CoNLL-2005

Proceedings of the

Ninth Conference on Computational Natural Language Learning

29-30 June 2005
University of Michigan
Ann Arbor, Michigan, USA

Production and Manufacturing by
Omnipress Inc.
Post Office Box 7214
Madison, WI 53707-7214

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
75 Paterson Street, Suite 9
New Brunswick, NJ 08901
USA
Tel: +1-732-342-9100
Fax: +1-732-342-9339
acl@aclweb.org

PREFACE

The 2005 Conference on Computational Natural Language Learning (CoNLL-2005) is the ninth in a series of meetings organized by SIGNLL, the ACL special interest group on natural language learning. This year's CoNLL will be held in Ann Arbor, Michigan, on June 29 and 30, in conjunction with the ACL 2005 conference.

This year we encouraged submissions addressing "deeper linguistic phenomena than have typically been covered in the past", a theme reflected in the shared task, and one which will be addressed by our invited speakers, Mark Johnson and Mark Steedman. The latter talk will be part of what should be a very interesting joint session with the Workshop on Psychocomputational Models of Human Language Acquisition organized by William Sakas.

A total of 70 papers were submitted to CoNLL's main session, of which we were able to accept only 19, making this the most competitive CoNLL meeting to date. We are particularly grateful to our program committee for their work reviewing this large number of papers on a tight schedule.

In keeping with the unique tradition of CoNLL, we also have a shared task session this year, on semantic role labeling. This year's task included use of full syntactic parses, a step beyond the chunk-level information used in 2004. The shared task was coordinated by Xavier Carreras and Lluís Màrquez. Common training and test data were made available to all participants, who had to build their learning system and test it on this task. The shared task also achieved a record number of submissions this year, and these proceedings include system descriptions from each of the 19 participants.

In addition to the program committee and shared task organizers, we are very indebted to the 2005 ACL conference organizers, in particular Dragomir Radev, Mirella Lapata, Jason Eisner, and Philipp Koehn, for their help with local arrangements and the publication of the proceedings.

We hope you enjoy this year's meeting!

Ido Dagan and Dan Gildea
May 2005

Organizers:

Ido Dagan, Bar-Ilan University (Israel)
Daniel Gildea, University of Rochester (USA)

Shared Task Organizers:

Xavier Carreras and Lluís Màrquez
Technical University of Catalonia (Spain)

Program Committee:

Steven Abney, University of Michigan (USA)
Eneko Agirre, University of the Basque Country (Spain)
Regina Barzilay, Massachusetts Institute of Technology (USA)
Claire Cardie, Cornell University (USA)
John Carroll, University of Sussex (UK)
Eugene Charniak, Brown University (USA)
James Cussens, University of York (UK)
Walter Daelemans, University of Antwerp (Belgium)
Radu Florian, IBM (USA)
Dayne Freitag, Fair Isaac (USA)
Rebecca Hwa, University of Pittsburgh (USA)
Hang Li, Microsoft (China)
Dekang Lin, University of Alberta (Canada)
Diane Litman, University of Pittsburgh (USA)
Yuji Matsumoto, Nara Institute of Science and Technology (Japan)
Diana McCarthy, University of Sussex (UK)
Rada Mihalcea, University of North Texas (USA)
John Nerbonne, University of Groningen (Netherlands)
Hwee-Tou Ng, National University of Singapore (Singapore)
Grace Ngai, The Hong Kong Polytechnic University (Hong Kong)
Miles Osborne, University of Edinburgh (UK)
Patrick Pantel, Information Sciences Institute (USA)
David Powers, Flinders University (Australia)
Dragomir Radev, University of Michigan (USA)
Ellen Riloff, University of Utah (USA)
Dan Roth, University of Illinois at Urbana-Champaign (USA)
Anoop Sarkar, Simon Fraser University (Canada)
Suzanne Stevenson, University of Toronto (Canada)
Keh Yih Su, Behavior Design Corporation (ROC)
Erik Tjong Kim Sang, University of Antwerp (Belgium)
Antal van den Bosch, Tilburg University (Netherlands)
Janyce Wiebe, University of Pittsburgh (USA)

Additional Reviewers:

Iñaki Alegria, Toine Bogers, Sander Canisius, Yunbo Cao, Hal Daume III, Guy De Pauw, Alex Fraser, Véronique Hoste, David Martinez, Art Munson, Vivi Năstase, Liang Zhou

Invited Speakers:

Mark Johnson, Brown University
Mark Steedman, University of Edinburgh

Table of Contents

Main Session

Conference Program

Wednesday, June 29, 2005

8:45 Welcome

Session 1: Syntax and Semantics

8:50 *Effective use of WordNet Semantics via Kernel-Based Learning*
Roberto Basili, Marco Cammisa and Alessandro Moschitti

9:15 *A Statistical Semantic Parser that Integrates Syntax and Semantics*
Ruifang Ge and Raymond Mooney

9:40 *Search Engine Statistics Beyond the n-gram: Application to Noun Compound Bracketing*
Preslav Nakov and Marti Hearst

10:05 *New Experiments in Distributional Representations of Synonymy*
Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer and Zhiqiang Wang

10:30 coffee break

Session 2: Semantics and Text Classification

11:00 *Word Independent Context Pair Classification Model for Word Sense Disambiguation*
Cheng Niu, Wei Li, Rohini K. Srihari and Huifeng Li

11:25 *Computing Word Similarity and Identifying Cognates with Pair Hidden Markov Models*
Wesley Mackay and Grzegorz Kondrak

11:50 *A Bayesian Mixture Model for Term Re-occurrence and Burstiness*
Avik Sarkar, Paul H. Garthwaite and Anne De Roeck

12:15 *Domain Kernels for Text Categorization*
Alfio Gliozzo and Carlo Strapparava

12:40 lunch

Wednesday, June 29, 2005 (continued)

Session 3: Information Extraction and Learning Methods

14:00 *Discriminative Training of Clustering Functions: Theory and Experiments with Entity Identification*
Xin Li and Dan Roth

14:25 *Using Uneven Margins SVM and Perceptron for Information Extraction*
Yaoyong Li, Kalina Bontcheva and Hamish Cunningham

14:50 *Improving Sequence Segmentation Learning by Predicting Trigrams*
Antal van den Bosch and Walter Daelemans

15:15 SIGNLL business meeting

15:40 coffee break

16:10 Invited Speaker: Mark Johnson

Session 4: Discourse and Anaphora

17:10 *An Expectation Maximization Approach to Pronoun Resolution*
Colin Cherry and Shane Bergsma

17:35 *Probabilistic Head-Driven Parsing for Discourse Structure*
Jason Baldridge and Alex Lascarides

Thursday June 30, 2005

Session 5: Joint Session with Workshop on Psychocomputational Models of Human Language Acquisition

9:00 Invited Speaker: Mark Steedman

9:50 *Steps Toward Deep Lexical Acquisition*
Sourabh Niyogi (Workshop on Psychocomputational Models)

10:15 *Intentional Context in Situated Natural Language Learning*
Michael Fleischman and Deb Roy

10:40 coffee break

Session 6: Morphology

11:10 *Representational Bias in Unsupervised Learning of Syllable Structure*
Sharon Goldwater and Mark Johnson

11:35 *An Analogical Learner for Morphological Analysis*
Nicolas Stroppa and François Yvon

12:00 *Morphology Induction from Term Clusters*
Dayne Freitag

12:25 lunch

Session 7: Learning Methods and Architectures

14:00 *Beyond the Pipeline: Discrete Optimization in NLP*
Tomasz Marciniak and Michael Strube

14:25 *Investigating the Effects of Selective Sampling on the Annotation Task*
Ben Hachey, Beatrice Alex and Markus Becker

14:50 **Shared Task:** Introduction and System Descriptions

15:30 coffee break

16:00 **Shared Task:** System Descriptions
17:00 **Shared Task:** Discussion
18:00 Closing

Effective use of WordNet semantics via kernel-based learning

Roberto Basili and **Marco Cammisa** and **Alessandro Moschitti**

Department of Computer Science
University of Rome "Tor Vergata", Rome, Italy
{basili,cammisa,moschitti}@info.uniroma2.it

Abstract

Research on document similarity has shown that complex representations are not more accurate than the simple *bag-of-words*. Term clustering, e.g. using latent semantic indexing, word co-occurrences or synonym relations using a word ontology have been shown not very effective. In particular, when to extend the similarity function external prior knowledge is used, e.g. WordNet, the retrieval system decreases its performance. The critical issues here are methods and conditions to integrate such knowledge.

In this paper we propose kernel functions to add prior knowledge to learning algorithms for document classification. Such kernels use a term similarity measure based on the WordNet hierarchy. The kernel trick is used to implement such space in a balanced and statistically coherent way. Cross-validation results show the benefit of the approach for the Support Vector Machines when few training data is available.

1 Introduction

The large literature on term clustering, term similarity and weighting schemes shows that document similarity is a central topic in Information Retrieval (IR). The research efforts have mostly been directed in enriching the document representation by using clustering (*term generalization*) or adding compounds (*term specifications*). These studies are based on the assumption that the similarity between two documents can be expressed as the similarity between pairs of matching terms. Following this idea, term clustering methods based on corpus term distributions or on external prior knowledge (e.g. provided by WordNet) were used to improve the basic term matching.

An example of statistical clustering is given in (Bekkerman et al., 2001). A feature selection technique, which clusters similar features/words, called the Information Bottleneck (IB), was applied to Text Categorization (TC). Such cluster based representation outperformed the simple *bag-of-words* on only one out of the three experimented collections. The effective use of external prior knowledge is even more difficult since no attempt has ever been successful to improve document retrieval or text classification accuracy, (e.g. see (Smeaton, 1999; Sussna, 1993; Voorhees, 1993; Voorhees, 1994; Moschitti and Basili, 2004)).

The main problem of term cluster based representations seems the unclear nature of the relationship between the word and the cluster information levels. Even if (semantic) clusters tend to improve the system recall, simple terms are, on a large scale, more accurate (e.g. (Moschitti and Basili, 2004)). To overcome this problem, hybrid spaces containing terms and clusters were experimented (e.g. (Scott and Matwin, 1999)) but the results, again, showed that the mixed statistical distributions of clusters and terms impact either marginally or even negatively on the overall accuracy.

In (Voorhees, 1993; Smeaton, 1999), clusters of synonymous terms as defined in WordNet (WN) (Fellbaum, 1998) were used for document retrieval. The results showed that the misleading information due to the wrong choice of the local term senses causes the overall accuracy to decrease. Word sense disambiguation (WSD) was thus applied beforehand by indexing the documents by means of disambiguated senses, i.e. synset codes (Smeaton, 1999;

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 1–8, Ann Arbor, June 2005.

Sussna, 1993; Voorhees, 1993; Voorhees, 1994; Moschitti and Basili, 2004). However, even the state-of-the-art methods for WSD did not improve the accuracy because of the inherent noise introduced by the disambiguation mistakes. The above studies suggest that term clusters decrease the precision of the system as they force weakly related or unrelated (in case of disambiguation errors) terms to give a contribution in the similarity function. The successful introduction of prior external knowledge relies on the solution of the above problem.

In this paper, a model to introduce the semantic lexical knowledge contained in the WN hierarchy in a supervised text classification task has been proposed. Intuitively, the main idea is that the documents d are represented through the set of all pairs in the vocabulary $< t, t' > \in V \times V$ originating by the terms $t \in d$ and all the words $t' \in V$, e.g. the WN nouns. When the similarity between two documents is evaluated, their matching pairs are used to account for the final score. The weight given to each term pair is proportional to the similarity that the two terms have in WN. Thus, the term t of the first document contributes to the document similarity according to its relatedness with any of the terms of the second document and the prior external knowledge, provided by WN, quantifies the single term to term relatedness. Such approach has two advantages: (a) we obtain a well defined space which supports the similarity between terms of different surface forms based on external knowledge and (b) we avoid to explicitly define term or sense clusters which inevitably introduce noise.

The class of spaces which embeds the above pair information may be composed by $O(|V|^2)$ dimensions. If we consider only the WN nouns (about 10^5), our space contains about 10^{10} dimensions which is not manageable by most of the learning algorithms. Kernel methods, can solve this problem as they allow us to use an implicit space representation in the learning algorithms. Among them Support Vector Machines (SVMs) (Vapnik, 1995) are kernel based learners which achieve high accuracy in presence of many irrelevant features. This is another important property as selection of the informative pairs is left to the SVM learning.

Moreover, as we believe that the prior knowledge in TC is not so useful when there is a sufficient amount of training documents, we experimented our model in poor training conditions (e.g. less equal than 20 documents for each category). The improvements in the accuracy, observed on the classification of the well known Reuters and 20 NewsGroups corpora, show that our document similarity model is very promising for general IR tasks: unlike previous attempts, it makes sense of the adoption of semantic external resources (i.e. WN) in IR.

Section 2 introduces the WordNet-based term similarity. Section 3 defines the new document similarity measure, the kernel function and its use within SVMs. Section 4 presents the comparative results between the traditional linear and the WN-based kernels within SVMs. In Section 5 comparative discussion against the related IR literature is carried out. Finally Section 6 derives the conclusions.

2 Term similarity based on general knowledge

In IR, any similarity metric in the vector space models is driven by lexical matching. When small training material is available, few words can be effectively used and the resulting document similarity metrics may be inaccurate. Semantic generalizations overcome data sparseness problems as contributions from different but semantically similar words are made available.

Methods for the induction of semantically inspired word clusters have been widely used in language modeling and lexical acquisition tasks (e.g. (Clark and Weir, 2002)). The resource employed in most works is WordNet (Fellbaum, 1998) which contains three subhierarchies: for nouns, verbs and adjectives. Each hierarchy represents lexicalized concepts (or senses) organized according to an *"is-a-kind-of"* relation. A concept s is described by a set of words $syn(s)$ called $synset$. The words $w \in syn(s)$ are synonyms according to the sense s.

For example, the words *line*, *argumentation*, *logical argument* and *line of reasoning* describe a synset which expresses the methodical process of logical reasoning (e.g. *"I can't follow your line of reasoning"*). Each word/term may be lexically related to more than one synset depending on its senses. The word *line* is also a member of the synset *line*, *dividing line*, *demarcation* and *contrast*, as a *line* denotes

also a conceptual separation (e.g. *"there is a narrow line between sanity and insanity*"). The Wordnet noun hierarchy is a direct acyclic graph[1] in which the edges establish the *direct_isa* relations between two synsets.

2.1 The Conceptual Density

The automatic use of WordNet for NLP and IR tasks has proved to be very complex. First, how the topological distance among senses is related to their corresponding conceptual distance is unclear. The pervasive lexical ambiguity is also problematic as it impacts on the measure of conceptual distances between word pairs. Second, the approximation of a set of concepts by means of their generalization in the hierarchy implies a conceptual loss that affects the target IR (or NLP) tasks. For example, *black* and *white* are *colors* but are also *chess pieces* and this impacts on the similarity score that should be used in IR applications. Methods to solve the above problems attempt to map a priori the terms to specific generalizations levels, i.e. to *cuts* in the hierarchy (e.g. (Li and Abe, 1998; Resnik, 1997)), and use corpus statistics for weighting the resulting mappings. For several tasks (e.g. in TC) this is unsatisfactory: different contexts of the same corpus (e.g. documents) may require different generalizations of the same word as they independently impact on the document similarity.

On the contrary, the *Conceptual Density* (CD) (Agirre and Rigau, 1996) is a flexible semantic similarity which depends on the generalizations of word senses not referring to any fixed level of the hierarchy. The CD defines a metrics according to the topological structure of WordNet and can be seemingly applied to two or more words. The measure formalized hereafter adapt to word pairs a more general definition given in (Basili et al., 2004).

We denote by $\bar{s}$ the set of nodes of the hierarchy rooted in the synset s, i.e. $\{c \in S | c\ isa\ s\}$, where S is the set of WN synsets. By definition $\forall s \in S, s \in \bar{s}$. CD makes a guess about the proximity of the senses, s_1 and s_2, of two words u_1 and u_2, according to the information expressed by the minimal sub-hierarchy, $\bar{s}$, that includes them. Let S_i be the set of generalizations for at least one sense s_i of the word u_i, i.e. $S_i = \{s \in S | s_i \in \bar{s}, u_i \in syn(s_i)\}$. The CD of u_1 and u_2 is:

$$CD(u_1, u_2) = \begin{cases} 0 & \textbf{iff } S_1 \cap S_2 = \emptyset \\ max_{s \in S_1 \cap S_2} \frac{\sum_{i=0}^{h} (\mu(\bar{s}))^i}{|\bar{s}|} & \textbf{otherwise} \end{cases} \quad (1)$$

where:

- $S_1 \cap S_2$ is the set of WN shared generalizations (i.e. the common hypernyms) of u_1 and u_2
- $\mu(\bar{s})$ is the average number of children per node (i.e. the branching factor) in the sub-hierarchy $\bar{s}$. $\mu(\bar{s})$ depends on WordNet and in some cases its value can approach 1.
- h is the depth of the *ideal*, i.e. maximally dense, *tree* with enough leaves to cover the two senses, s_1 and s_2, according to an average branching factor of $\mu(\bar{s})$. This value is actually estimated by:

$$h = \begin{cases} \lfloor log_{\mu(\bar{s})} 2 \rfloor & \textbf{iff } \mu(\bar{s}) \neq 1 \\ 2 & \textbf{otherwise} \end{cases} \quad (2)$$

  When $\mu(s)$=1, h ensures a tree with at least 2 nodes to cover s_1 and s_2 ($height = 2$).
- $|\bar{s}|$ is the number of nodes in the sub-hierarchy $\bar{s}$. This value is statically measured on WN and it is a negative bias for the higher level generalizations (i.e. larger $\bar{s}$).

CD models the semantic distance as the density of the generalizations $s \in S_1 \cap S_2$. Such *density* is the ratio between the number of nodes of the *ideal tree* and $|\bar{s}|$. The ideal tree should (a) link the two senses/nodes s_1 and s_2 with the minimal number of edges (isa-relations) and (b) maintain the same branching factor (*bf*) observed in $\bar{s}$. In other words, this tree provides the minimal number of nodes (and isa-relations) sufficient to connect s_1 and s_2 according to the topological structure of $\bar{s}$. For example, if $\bar{s}$ has a *bf* of 2 the ideal tree connects the two senses with a single node (their father). If the *bf* is 1.5, to replicate it, the ideal tree must contain 4 nodes, i.e. the grandfather which has a *bf* of 1 and the father which has *bf* of 2 for an average of 1.5. When *bf* is 1 the Eq. 1 degenerates to the inverse of the number of nodes in the path between s_1 and s_2, i.e. the simple proximity measure used in (Siolas and d'Alch Buc, 2000).

[1] As only the 1% of its nodes own more than one parent in the graph, most of the techniques assume the hierarchy to be a tree, and treat the few exception heuristically.

It is worth noting that for each pair $CD(u_1, u_2)$ determines the similarity according to *the closest lexical senses*, s_1, $s_2 \in \bar{s}$: the remaining senses of u_1 and u_2 are irrelevant, with a resulting semantic disambiguation side effect. CD has been successfully applied to semantic tagging ((Basili et al., 2004)). As the WN hierarchies for other POS classes (i.e. verb and adjectives) have topological properties different from the noun hyponimy network, their semantics is not suitably captured by Eq. 1. In this paper, Eq. 1 has thus been only applied to noun pairs. As the high number of such pairs increases the computational complexity of the target learning algorithm, efficient approaches are needed. The next section describes how kernel methods can make practical the use of the Conceptual Density in Text Categorization.

3 A WordNet Kernel for document similarity

Term similarities are used to design document similarities which are the core functions of most TC algorithms. The term similarity proposed in Eq. 1 is valid for all term pairs of a target vocabulary and has two main advantages: (1) the relatedness of each term occurring in the first document can be computed against *all* terms in the second document, i.e. all different pairs of similar (not just identical) tokens can contribute and (2) if we use all term pair contributions in the document similarity we obtain a measure consistent with the term probability distributions, i.e. the sum of all term contributions does not penalize or emphasize arbitrarily any subset of terms. The next section presents more formally the above idea.

3.1 A *semantic* vector space

Given two documents d_1 and $d_2 \in D$ (the document-set) we define their similarity as:

$$K(d_1, d_2) = \sum_{w_1 \in d_1, w_2 \in d_2} (\lambda_1 \lambda_2) \times \sigma(w_1, w_2) \quad (3)$$

where λ_1 and λ_2 are the weights of the words (features) w_1 and w_2 in the documents d_1 and d_2, respectively and σ is a term similarity function, e.g. the conceptual density defined in Section 2. To prove that Eq. 3 is a valid kernel is enough to show that it is a specialization of the general definition of convolution kernels formalized in (Haussler, 1999). Hereafter, we report such definition. Let $X, X_1, .., X_m$ be separable metric spaces, $x \in X$ a structure and $\vec{x} = x_1, ..., x_m$ its parts, where $x_i \in X_i \quad \forall i = 1, .., m$. Let R be a relation on the set $X \times X_1 \times .. \times X_m$ such that $R(\vec{x}, x)$ is "true" if $\vec{x}$ are the parts of x. We indicate with $R^{-1}(x)$ the set $\{\vec{x} : R(\vec{x}, x)\}$. Given two objects x and $y \in X$ their similarity $K(x, y)$ is defined as:

$$K(x, y) = \sum_{\vec{x} \in R^{-1}(x)} \sum_{\vec{y} \in R^{-1}(y)} \prod_{i=1}^{m} K_i(x_i, y_i) \quad (4)$$

If X defines the document set (i.e. $D = X$), and X_1 the vocabulary of the target document corpus ($X_1 = V$), it follows that: $x = d$ (a document), $\vec{x} = x_1 = w \in V$ (a word which is a part of the document d) and $R^{-1}(d)$ defines the set of words in the document d. As $\prod_{i=1}^{m} K_i(x_i, y_i) = K_1(x_1, y_1)$, then $K_1(x_1, y_1) = K(w_1, w_2) = (\lambda_1 \lambda_2) \times \sigma(w_1, w_2)$, i.e. Eq. 3.

The above equation can be used in support vector machines as illustrated by the next section.

3.2 Support Vector Machines and Kernel methods

Given the vector space in $\mathbb{R}^\eta$ and a set of positive and negative points, SVMs classify vectors according to a separating hyperplane, $H(\vec{x}) = \vec{\omega} \cdot \vec{x} + b = 0$, where $\vec{x}$ and $\vec{\omega} \in \mathbb{R}^\eta$ and $b \in \mathbb{R}$ are learned by applying the *Structural Risk Minimization principle* (Vapnik, 1995). From the kernel theory we have that:

$$H(\vec{x}) = \Big(\sum_{h=1..l} \alpha_h \vec{x_h} \Big) \cdot \vec{x} + b = \sum_{h=1..l} \alpha_h \vec{x}_h \cdot \vec{x} + b =$$
$$\sum_{h=1..l} \alpha_h \phi(d_h) \cdot \phi(d) + b = \sum_{h=1..l} \alpha_h K(d_h, d) + b \quad (5)$$

where, d is a classifying document and d_h are all the l training instances, projected in $\vec{x}$ and $\vec{x}_h$ respectively. The product $K(d, d_h) = <\phi(d) \cdot \phi(d_h)>$ is the *Semantic WN-based Kernel* (SK) function associated with the mapping ϕ.

Eq. 5 shows that to evaluate the separating hyperplane in $\mathbb{R}^\eta$ we do not need to evaluate the entire vector $\vec{x_h}$ or $\vec{x}$. Actually, we do not know even the mapping ϕ and the number of dimensions, η. As it is sufficient to compute $K(d, d_h)$, we can carry out the learning with Eq. 3 in the $\mathbb{R}^n$, avoiding to

use the explicit representation in the $\mathbb{R}^\eta$ space. The real advantage is that we can consider only the word pairs associated with non-zero weight, i.e. we can use a sparse vector computation. Additionally, to have a uniform score across different document size, the kernel function can be normalized as follows: $\frac{SK(d_1,d_2)}{\sqrt{SK(d_1,d_1)\cdot SK(d_2,d_2)}}$

4 Experiments

The use of WordNet (WN) in the term similarity function introduces a prior knowledge whose impact on the Semantic Kernel (SK) should be experimentally assessed. The main goal is to compare the traditional Vector Space Model kernel against SK, both within the Support Vector learning algorithm.

The high complexity of the SK limits the size of the experiments that we can carry out in a feasible time. Moreover, we are not interested to large collections of training documents as in these training conditions the simple *bag-of-words* models are in general very effective, i.e. they seems to model well the document similarity needed by the learning algorithms. Thus, we carried out the experiments on small subsets of the 20NewsGroups[2] (20NG) and the *Reuters-21578*[3] corpora to simulate critical learning conditions.

4.1 Experimental set-up

For the experiments, we used the SVM-light software (Joachims, 1999) (available at `svmlight.joachims.org`) with the default linear kernel on the token space (adopted as the baseline evaluations). For the SK evaluation we implemented the Eq. 3 with $\sigma(\cdot,\cdot) = CD(\cdot,\cdot)$ (Eq. 1) inside SVM-light. As Eq. 1 is only defined for nouns, a part of speech (POS) tagger has been previously applied. However, also verbs, adjectives and numerical features were included in the pair space. For these tokens a $CD = 0$ is assigned to pairs made by different strings. As the POS-tagger could introduce errors, in a second experiment, any token with a successful look-up in the WN noun hierarchy was considered in the kernel. This approximation has the benefit to retrieve useful information even for verbs and capture the similarity between verbs and some nouns, e.g. *to drive* (via the noun *drive*) has a common synset with *parkway*.

For the evaluations, we applied a careful SVM parameterization: a preliminary investigation suggested that the trade-off (between the training-set error and margin, i.e. c option in SVM-light) parameter optimizes the F_1 measure for values in the range [0.02,0.32][4]. We noted also that the cost-factor parameter (i.e. j option) is not critical, i.e. a value of 10 always optimizes the accuracy. The feature selection techniques and the weighting schemes were not applied in our experiments as they cannot be accurately estimated from the small available training data.

The classification performance was evaluated by means of the F_1 measure[5] for the single category and the MicroAverage for the final classifier pool (Yang, 1999). Given the high computational complexity of SK we selected 8 categories from the 20NG[6] and 8 from the Reuters corpus[7]

To derive statistically significant results with few training documents, for each corpus, we randomly selected 10 different samples from the 8 categories. We trained the classifiers on one sample, parameterized on a second sample and derived the measures on the other 8. By rotating the training sample we obtained 80 different measures for each model. The size of the samples ranges from 24 to 160 documents depending on the target experiment.

4.2 Cross validation results

The SK (Eq. 3) was compared with the linear kernel which obtained the best F_1 measure in (Joachims, 1999). Table 1 reports the first comparative results for 8 categories of 20NG on 40 training documents. The results are expressed as the *Mean* and the *Std. Dev.* over 80 runs. The F_1 are reported in Column 2 for the linear kernel, i.e. bow, in Column 3 for SK without applying POS information and in Column 4

[2] Available at `www.ai.mit.edu/people/jrennie/20Newsgroups/`.

[3] The Apté split available at `kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

[4] We used all the values from 0.02 to 0.32 with step 0.02.

[5] F_1 assigns equal importance to Precision P and Recall R, i.e. $F_1 = \frac{2P\cdot R}{P+R}$.

[6] We selected the 8 most different categories (in terms of their content) i.e. *Atheism*, *Computer Graphics*, *Misc Forsale*, *Autos*, *Sport Baseball*, *Medicine*, *Talk Religions* and *Talk Politics*.

[7] We selected the 8 largest categories, i.e. *Acquisition*, *Earn*, *Crude*, *Grain*, *Interest*, *Money-fx*, *Trade* and *Wheat*.

for SK with the use of POS information (SK-POS). The last row shows the MicroAverage performance for the above three models on all 8 categories. We note that SK improves bow of 3%, i.e. 34.3% vs. 31.5% and that the POS information reduces the improvement of SK, i.e. 33.5% vs. 34.3%.

To verify the hypothesis that WN information is useful in low training data conditions we repeated the evaluation over the 8 categories of Reuters with samples of 24 and 160 documents, respectively. The results reported in Table 2 shows that (1) again SK improves bow (41.7% - 37.2% = 4.5%) and (2) as the number of documents increases the improvement decreases (77.9% - 75.9% = 2%). It is worth noting that the standard deviations tend to assume high values. In general, the use of 10 disjoint training/testing samples produces a higher variability than the n-fold cross validation which insists on the same document set. However, this does not affect the t-student confidence test over the differences between the MicroAverage of SK and *bow* since the former has a higher accuracy at 99% confidence level.

The above findings confirm that SK outperforms the *bag-of-words* kernel in critical learning conditions as the semantic contribution of the SK recovers useful information. To complete this study we carried out experiments with samples of different size, i.e. 3, 5, 10, 15 and 20 documents for each category. Figures 1 and 2 show the learning curves for 20NG and Reuters corpora. Each point refers to the average on 80 samples.

As expected the improvement provided by SK decreases when more training data is available. However, the improvements are not negligible yet. The SK model (without POS information) preserves about 2-3% of improvement with 160 training documents. The matching allowed between noun-verb pairs still captures semantic information which is useful for topic detection. In particular, during the similarity estimation, each word activates 60.05 pairs on average. This is particularly useful to increase the amount of information available to the SVMs.

Finally, we carried out some experiments with 160 Reuters documents by discarding the string matching from SK. Only words having different surface forms were allowed to give contributions to the Eq. 3.

Category	bow	SK	SK-POS
Atheism	29.5±19.8	32.0±16.3	25.2±17.2
Comp.Graph	39.2±20.7	39.3±20.8	29.3±21.8
Misc.Forsale	61.3±17.7	51.3±18.7	49.5±20.4
Autos	26.2±22.7	26.0±20.6	33.5±26.8
Sport.Baseb.	32.7±20.1	36.9±22.5	41.8±19.2
Sci.Med	26.1±17.2	18.5±17.4	16.6±17.2
Talk.Relig.	23.5±11.6	28.4±19.0	27.6±17.0
Talk.Polit.	28.3±17.5	30.7±15.5	30.3±14.3
MicroAvg. F_1	31.5±4.8	34.3±5.8	33.5±6.4

Table 1: Performance of the linear and Semantic Kernel with 40 training documents over 8 categories of 20NewsGroups collection.

Category	24 docs		160 docs	
	bow	SK	bow	SK
Acq.	55.3±18.1	50.8±18.1	86.7±4.6	84.2±4.3
Crude	3.4±5.6	3.5±5.7	64.0±20.6	62.0±16.7
Earn	64.0±10.0	64.7±10.3	91.3±5.5	90.4±5.1
Grain	45.0±33.4	44.4±29.6	69.9±16.3	73.7±14.8
Interest	23.9±29.9	24.9±28.6	67.2±12.9	59.8±12.6
Money-fx	36.1±34.3	39.2±29.5	69.1±11.9	67.4±13.3
Trade	9.8±21.2	10.3±17.9	57.1±23.8	60.1±15.4
Wheat	8.6±19.7	13.3±26.3	23.9±24.8	31.2±23.0
Mic.Avg.	37.2±5.9	41.7±6.0	75.9±11.0	77.9±5.7

Table 2: Performance of the linear and Semantic Kernel with 40 and 160 training documents over 8 categories of the Reuters corpus.

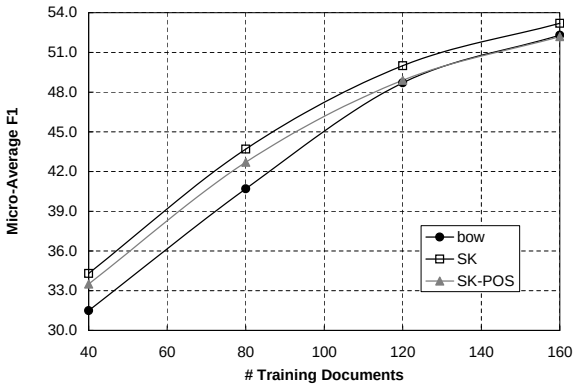


Figure 1: MicroAverage F_1 of SVMs using bow, SK and SK-POS kernels over the 8 categories of 20NewsGroups.

The important outcome is that SK converges to a MicroAverage F_1 measure of 56.4% (compare with Table 2). This shows that the word similarity provided by WN is still consistent and, although in the worst case, slightly effective for TC: the evidence is that a suitable balancing between lexical ambiguity and topical relatedness is captured by the SVM learning.

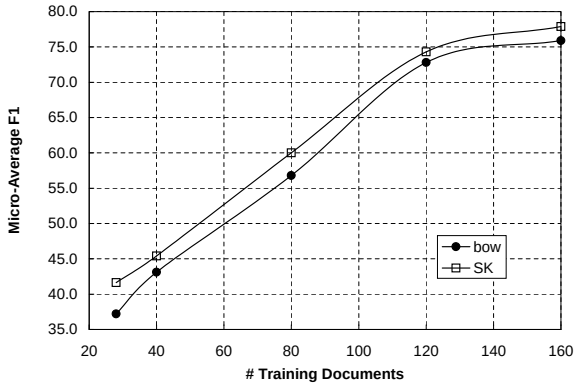


Figure 2: MicroAverage F_1 of SVMs using bow and SK over the 8 categories of the Reuters corpus.

5 Related Work

The IR studies in this area focus on the term similarity models to embed statistical and external knowledge in document similarity.

In (Kontostathis and Pottenger, 2002) a *Latent Semantic Indexing* analysis was used for term clustering. Such approach assumes that values x_{ij} in the transformed term-term matrix represents the similarity (> 0) and anti-similarity between terms i and j. By extension, a negative value represents an anti-similarity between i and j enabling both positive and negative clusters of terms. Evaluation of query expansion techniques showed that positive clusters can improve Recall of about 18% for the *CISI* collection, 2.9% for *MED* and 3.4% for *CRAN*. Furthermore, the negative clusters, when used to prune the result set, improve the precision.

The use of external semantic knowledge seems to be more problematic in IR. In (Smeaton, 1999), the impact of semantic ambiguity on IR is studied. A WN-based semantic similarity function between noun pairs is used to improve indexing and document-query matching. However, the WSD algorithm had a performance ranging between 60-70%, and this made the overall semantic similarity not effective.

Other studies using semantic information for improving IR were carried out in (Sussna, 1993) and (Voorhees, 1993; Voorhees, 1994). Word semantic information was here used for text indexing and query expansion, respectively. In (Voorhees, 1994) it is shown that semantic information derived directly from WN without a priori WSD produces poor results.

The latter methods are even more problematic in TC (Moschitti and Basili, 2004). Word senses tend to systematically correlate with the positive examples of a category. Different categories are better characterized by different words rather than different senses. Patterns of lexical co-occurrences in the training data seem to suffice for automatic disambiguation. (Scott and Matwin, 1999) use WN senses to replace simple words without word sense disambiguation and small improvements are derived only for a small corpus. The scale and assessment provided in (Moschitti and Basili, 2004) (3 corpora using cross-validation techniques) showed that even the accurate disambiguation of WN senses (about 80% accuracy on nouns) did not improve TC.

In (Siolas and d'Alch Buc, 2000) was proposed an approach similar to the one presented in this article. A term proximity function is used to design a kernel able to semantically smooth the similarity between two document terms. Such semantic kernel was designed as a combination of the Radial Basis Function (RBF) kernel with the term proximity matrix. Entries in this matrix are inversely proportional to the length of the WN hierarchy path linking the two terms. The performance, measured over the 20NewsGroups corpus, showed an improvement of 2% over the *bag-of-words*. Three main differences exist with respect to our approach. First, the term proximity does not fully capture the WN topological information. Equidistant terms receive the same similarity irrespectively from their generalization level. For example, *Sky* and *Location* (direct hyponyms of *Entity*) receive a similarity score equal to *knife* and *gun* (hyponyms of *weapon*). More accurate measures have been widely discussed in literature, e.g. (Resnik, 1997). Second, the kernel-based CD similarity is an elegant combination of lexicalized and semantic information. In (Siolas and d'Alch Buc, 2000) the combination of weighting schemes, the RBF kernel and the proximitry matrix has a much less clear interpretation. Finally, (Siolas and d'Alch Buc, 2000) selected only 200 features via Mutual Information statistics. In this way rare or non statistically significant terms are neglected while being source of often relevant contributions in the SK space modeled over WN.

Other important work on semantic kernel for retrieval has been developed in (Cristianini et al.,

2002; Kandola et al., 2002). Two methods for inferring semantic similarity from a corpus were proposed. In the first a system of equations were derived from the dual relation between word-similarity based on document-similarity and viceversa. The equilibrium point was used to derive the semantic similarity measure. The second method models semantic relations by means of a diffusion process on a graph defined by lexicon and co-occurrence information. The major difference with our approach is the use of a different source of prior knowledge. Similar techniques were also applied in (Hofmann, 2000) to derive a Fisher kernel based on a latent class decomposition of the term-document matrix.

6 Conclusions

The introduction of semantic prior knowledge in IR has always been an interesting subject as the examined literature suggests. In this paper, we used the conceptual density function on the WordNet (WN) hierarchy to define a document similarity metric. Accordingly, we defined a semantic kernel to train Support Vector Machine classifiers. Cross-validation experiments over 8 categories of 20NewsGroups and Reuters over multiple samples have shown that in poor training data conditions, the WN prior knowledge can be effectively used to improve (up to 4.5 absolute percent points, i.e. 10%) the TC accuracy.

These promising results enable a number of future researches: (1) larger scale experiments with different measures and semantic similarity models (e.g. (Resnik, 1997)); (2) improvement of the overall efficiency by exploring feature selection methods over the SK, and (3) the extension of the semantic similarity by a general (i.e. non binary) application of the conceptual density model.

References

E. Agirre and G. Rigau. 1996. Word sense disambiguation using conceptual density. In *Proceedings of COLING'96, Copenhagen, Danmark.*

R. Basili, M. Cammisa, and F. M. Zanzotto. 2004. A similarity measure for unsupervised semantic disambiguation. In *In Proceedings of Language Resources and Evaluation Conference*, Lisbon, Portugal.

Ron Bekkerman, Ran El-Yaniv, Naftali Tishby, and Yoad Winter. 2001. On feature distributional clustering for text categorization. In *Proceedings of SIGIR'01* , New Orleans, Louisiana, US.

Stephen Clark and David Weir. 2002. Class-based probability estimation using a semantic hierarchy. *Comput. Linguist.*, 28(2):187–206.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. 2002. Latent semantic kernels. *J. Intell. Inf. Syst.*, 18(2-3):127–152.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

D. Haussler. 1999. Convolution kernels on discrete structures. Technical report ucs-crl-99-10, University of California Santa Cruz.

Thomas Hofmann. 2000. Learning probabilistic models of the web. In *Research and Development in Information Retrieval*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

J. Kandola, J. Shawe-Taylor, and N. Cristianini. 2002. Learning semantic similarity. In *NIPS'02) - MIT Press.*

A. Kontostathis and W. Pottenger. 2002. Improving retrieval performance with positive and negative equivalence classes of terms.

Hang Li and Naoki Abe. 1998. Generalizing case frames using a thesaurus and the mdl principle. *Computational Linguistics*, 23(3).

Alessandro Moschitti and Roberto Basili. 2004. Complex linguistic features for text classification: a comprehensive study. In *Proceedings of ECIR'04*, Sunderland, UK.

P. Resnik. 1997. Selectional preference and sense disambiguation. In *Proceedings of ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?, Washington, 1997.*

Sam Scott and Stan Matwin. 1999. Feature engineering for text classification. In *Proceedings of ICML'99*, Bled, SL. Morgan Kaufmann Publishers, San Francisco, US.

Georges Siolas and Florence d'Alch Buc. 2000. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of IJCNN'00*. IEEE Computer Society.

Alan F. Smeaton. 1999. Using NLP or NLP resources for information retrieval tasks. In *Natural language information retrieval*, Kluwer Academic Publishers, Dordrecht, NL.

M. Sussna. 1993. Word sense disambiguation for free-text indexing using a massive semantic network. In *CKIM'93,*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Ellen M. Voorhees. 1993. Using wordnet to disambiguate word senses for text retrieval. In *Proceedings SIGIR'93* Pittsburgh, PA, USA.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of SIGIR'94*, ACM/Springer.

Y. Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval Journal*.

A Statistical Semantic Parser that Integrates Syntax and Semantics

Ruifang Ge Raymond J. Mooney
Department of Computer Sciences
University of Texas, Austin
TX 78712, USA
{grf,mooney}@cs.utexas.edu

Abstract

We introduce a learning semantic parser, SCISSOR, that maps natural-language sentences to a detailed, formal, meaning-representation language. It first uses an integrated statistical parser to produce a semantically augmented parse tree, in which each non-terminal node has both a syntactic and a semantic label. A compositional-semantics procedure is then used to map the augmented parse tree into a final meaning representation. We evaluate the system in two domains, a natural-language database interface and an interpreter for coaching instructions in robotic soccer. We present experimental results demonstrating that SCISSOR produces more accurate semantic representations than several previous approaches.

1 Introduction

Most recent work in learning for semantic parsing has focused on "shallow" analysis such as *semantic role labeling* (Gildea and Jurafsky, 2002). In this paper, we address the more ambitious task of learning to map sentences to a complete formal *meaning-representation language* (MRL). We consider two MRL's that can be directly used to perform useful, complex tasks. The first is a Prolog-based language used in a previously-developed corpus of queries to a database on U.S. geography (Zelle and Mooney, 1996). The second MRL is a coaching language for robotic soccer developed for the RoboCup Coach Competition, in which AI researchers compete to provide effective instructions to a coachable team of agents in a simulated soccer domain (et al., 2003).

We present an approach based on a statistical parser that generates a *semantically augmented parse tree* (SAPT), in which each internal node includes both a syntactic and semantic label. We augment Collins' head-driven model 2 (Collins, 1997) to incorporate a semantic label on each internal node. By integrating syntactic and semantic interpretation into a single statistical model and finding the globally most likely parse, an accurate combined syntactic/semantic analysis can be obtained. Once a SAPT is generated, an additional step is required to translate it into a final formal *meaning representation* (MR).

Our approach is implemented in a system called SCISSOR (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations). Training the system requires sentences annotated with both gold-standard SAPT's and MR's. We present experimental results on corpora for both geography-database querying and Robocup coaching demonstrating that SCISSOR produces more accurate semantic representations than several previous approaches based on symbolic learning (Tang and Mooney, 2001; Kate et al., 2005).

2 Target MRL's

We used two MRLs in our experiments: CLANG and GEOQUERY. They capture the meaning of linguistic utterances in their domain in a formal language.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 9–16, Ann Arbor, June 2005.

2.1 CLANG: the RoboCup Coach Language

RoboCup (www.robocup.org) is an international AI research initiative using robotic soccer as its primary domain. In the Coach Competition, teams of agents compete on a simulated soccer field and receive advice from a team coach in a formal language called CLANG. In CLANG, tactics and behaviors are expressed in terms of if-then rules. As described in (et al., 2003), its grammar consists of 37 non-terminal symbols and 133 productions. Below is a sample rule with its English gloss:

```
((bpos (penalty-area our))
 (do (player-except our {4})
     (pos (half our))))
```

"If the ball is in our penalty area, all our players except player 4 should stay in our half."

2.2 GEOQUERY: a DB Query Language

GEOQUERY is a logical query language for a small database of U.S. geography containing about 800 facts. This domain was originally chosen to test corpus-based semantic parsing due to the availability of a hand-built natural-language interface, GEOBASE, supplied with Turbo Prolog 2.0 (Borland International, 1988). The GEOQUERY language consists of Prolog queries augmented with several meta-predicates (Zelle and Mooney, 1996). Below is a sample query with its English gloss:

```
answer(A,count(B,(city(B),loc(B,C),
       const(C,countryid(usa))),A))
```

"How many cities are there in the US?"

3 Semantic Parsing Framework

This section describes our basic framework for semantic parsing, which is based on a fairly standard approach to compositional semantics (Jurafsky and Martin, 2000). First, a statistical parser is used to construct a SAPT that captures the semantic interpretation of individual words and the basic predicate-argument structure of the sentence. Next, a recursive procedure is used to compositionally construct an MR for each node in the SAPT from the semantic label of the node and the MR's

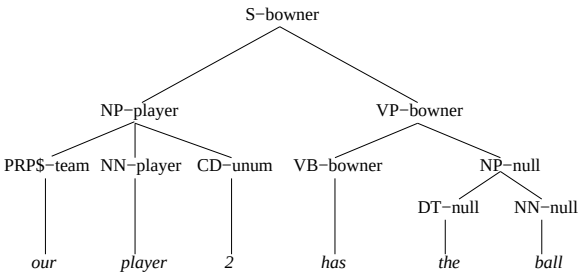


Figure 1: An SAPT for a simple CLANG sentence.

Function: BUILDMR(N, K)
Input: The root node N of a SAPT;
predicate-argument knowledge, K, for the MRL.
Notation: X_{MR} is the MR of node X.
Output: N_{MR}
$C_i :=$ the ith child node of $N, 1 \leq i \leq n$
$C_h =$ GETSEMANTICHEAD(N) // see Section 3
$C_{h_{MR}} =$ BUILDMR(C_h, K)
for each other child C_i where $i \neq h$
$C_{i_{MR}} =$ BUILDMR(C_i, K)
COMPOSEMR$(C_{h_{MR}}, C_{i_{MR}}, K)$ // see Section 3
$N_{MR} = C_{h_{MR}}$

Figure 2: Computing an MR from a SAPT.

of its children. Syntactic structure provides information of how the parts should be composed. Ambiguities arise in both syntactic structure and the semantic interpretation of words and phrases. By integrating syntax and semantics in a single statistical parser that produces an SAPT, we can use both semantic information to resolve syntactic ambiguities and syntactic information to resolve semantic ambiguities.

In a SAPT, each internal node in the parse tree is annotated with a semantic label. Figure 1 shows the SAPT for a simple sentence in the CLANG domain. The semantic labels which are shown after dashes are *concepts* in the domain. Some *type concepts* do not take arguments, like *team* and *unum* (uniform number). Some concepts, which we refer to as *predicates*, take an ordered list of arguments, like *player* and *bowner* (ball owner). The predicate-argument knowledge, K, specifies, for each predicate, the semantic constraints on its arguments. Constraints are specified in terms of the concepts that can fill each argument, such as *player*(*team*, *unum*) and *bowner*(*player*). A special semantic label $null$ is used for nodes that do not correspond to any concept in the domain.

Figure 2 shows the basic algorithm for building an MR from an SAPT. Figure 3 illustrates the

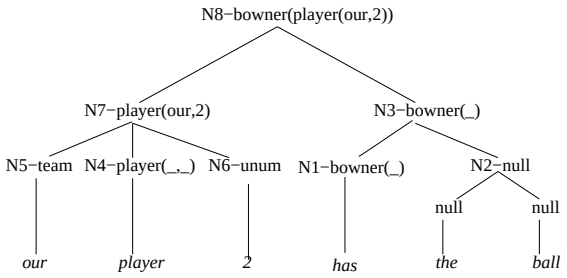


Figure 3: MR's constructed for each SAPT Node.

construction of the MR for the SAPT in Figure 1. Nodes are numbered in the order in which the construction of their MR's are completed. The first step, GETSEMANTICHEAD, determines which of a node's children is its *semantic head* based on having a matching semantic label. In the example, node N3 is determined to be the semantic head of the sentence, since its semantic label, *bowner*, matches N8's semantic label. Next, the MR of the semantic head is constructed recursively. The semantic head of N3 is clearly N1. Since N1 is a part-of-speech (POS) node, its semantic label directly determines its MR, which becomes *bowner*(_). Once the MR for the head is constructed, the MR of all other (non-head) children are computed recursively, and COMPOSEMR assigns their MR's to fill the arguments in the head's MR to construct the complete MR for the node. Argument constraints are used to determine the appropriate filler for each argument. Since, N2 has a *null* label, the MR of N3 also becomes *bowner*(_). When computing the MR for N7, N4 is determined to be the head with the MR: *player*(_,_). COMPOSEMR then assigns N5's MR to fill the *team* argument and N6's MR to fill the *unum* argument to construct N7's complete MR: *player*(*our*, *2*). This MR in turn is composed with the MR for N3 to yield the final MR for the sentence: *bowner*(*player*(*our*,*2*)).

For MRL's, such as CLANG, whose syntax does not strictly follow a nested set of predicates and arguments, some final minor syntactic adjustment of the final MR may be needed. In the example, the final MR is *(bowner (player our {2}))*. In the following discussion, we ignore the difference between these two.

There are a few complications left which require special handling when generating MR's, like coordination, anaphora resolution and non-compositionality exceptions. Due to space limitations, we do not present the straightforward techniques we used to handle them.

4 Corpus Annotation

This section discusses how sentences for training SCISSOR were manually annotated with SAPT's. Sentences were parsed by Collins' head-driven model 2 (Bikel, 2004) (trained on sections 02-21 of the WSJ Penn Treebank) to generate an initial syntactic parse tree. The trees were then manually corrected and each node augmented with a semantic label.

First, semantic labels for individual words, called *semantic tags*, are added to the POS nodes in the tree. The tag *null* is used for words that have no corresponding concept. Some concepts are conveyed by phrases, like "has the ball" for *bowner* in the previous example. Only one word is labeled with the concept; the syntactic head word (Collins, 1997) is preferred. During parsing, the other words in the phrase will provide context for determining the semantic label of the head word.

Labels are added to the remaining nodes in a bottom-up manner. For each node, one of its children is chosen as the semantic head, from which it will inherit its label. The semantic head is chosen as the child whose semantic label can take the MR's of the other children as arguments. This step was done mostly automatically, but required some manual corrections to account for unusual cases.

In order for COMPOSEMR to be able to construct the MR for a node, the argument constraints for its semantic head must identify a unique concept to fill each argument. However, some predicates take multiple arguments of the same type, such as *point.num*(*num*,*num*), which is a kind of point that represents a field coordinate in CLANG.

In this case, extra nodes are inserted in the tree with new type concepts that are unique for each argument. An example is shown in Figure 4 in which the additional type concepts *num1* and *num2* are introduced. Again, during parsing, context will be used to determine the correct type for a given word.

The *point* label of the root node of Figure 4 is the concept that includes all kinds of points in CLANG. Once a predicate has all of its arguments filled, we

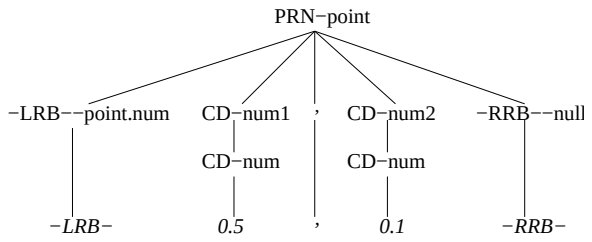


Figure 4: Adding new types to disambiguate arguments.

use the most general CLANG label for its concept (e.g. *point* instead of *point.num*). This generality avoids sparse data problems during training.

5 Integrated Parsing Model

5.1 Collins Head-Driven Model 2

Collins' head-driven model 2 is a generative, lexicalized model of statistical parsing. In the following section, we follow the notation in (Collins, 1997). Each non-terminal X in the tree is a syntactic label, which is lexicalized by annotating it with a *word*, w, and a *POS tag*, t_{syn}. Thus, we write a non-terminal as $X(x)$, where X is a syntactic label and $x = \langle w, t_{syn} \rangle$. $X(x)$ is then what is generated by the generative model.

Each production $LHS \Rightarrow RHS$ in the PCFG is in the form:

$$P(h) \rightarrow L_n(l_n)...L_1(l_1)H(h)R_1(r_1)...R_m(r_m)$$

where H is the head-child of the phrase, which inherits the head-word h from its parent P. $L_1...L_n$ and $R_1...R_m$ are left and right modifiers of H.

Sparse data makes the direct estimation of $\mathcal{P}(RHS|LHS)$ infeasible. Therefore, it is decomposed into several steps – first generating the head, then the right modifiers from the head outward, then the left modifiers in the same way. Syntactic subcategorization frames, LC and RC, for the left and right modifiers respectively, are generated before the generation of the modifiers. Subcat frames represent knowledge about subcategorization preferences. The final probability of a production is composed from the following probabilities:

1. The probability of choosing a head constituent label H: $\mathcal{P}_h(H|P,h)$.
2. The probabilities of choosing the left and right subcat frames LC and RC: $\mathcal{P}_{lc}(LC|P,H,h)$ and $\mathcal{P}_{rc}(RC|P,H,h)$.

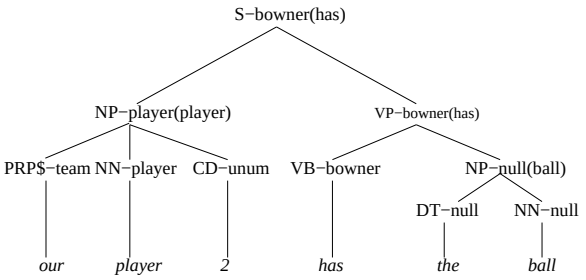


Figure 5: A lexicalized SAPT.

3. The probabilities of generating the left and right modifiers: $\prod_{i=1..m+1} \mathcal{P}_r(R_i(r_i)|H,P,h,\Delta_{i-1},RC) \times \prod_{i=1..n+1} \mathcal{P}_l(L_i(l_i)|H,P,h,\Delta_{i-1},LC)$. Where Δ is the measure of the distance from the head word to the edge of the constituent, and $L_{n+1}(l_{n+1})$ and $R_{m+1}(r_{m+1})$ are $STOP$. The model stops generating more modifiers when $STOP$ is generated.

5.2 Integrating Semantics into the Model

We extend Collins' model to include the generation of semantic labels in the derivation tree. Unless otherwise stated, notation has the same meaning as in Section 5.1. The subscript syn refers to the syntactic part, and sem refers to the semantic part. We redefine X and x to include semantics, each non-terminal X is now a pair of a syntactic label X_{syn} and a semantic label X_{sem}. Besides being annotated with the word, w, and the POS tag, t_{syn}, X is also annotated with the semantic tag, t_{sem}, of the head child. Thus, $X(x)$ now consists of $X = \langle X_{syn}, X_{sem} \rangle$, and $x = \langle w, t_{syn}, t_{sem} \rangle$. Figure 5 shows a lexicalized SAPT (but omitting t_{syn} and t_{sem}).

Similar to the syntactic subcat frames, we also condition the generation of modifiers on semantic subcat frames. Semantic subcat frames give semantic subcategorization preferences; for example, *player* takes a *team* and a *unum*. Thus LC and RC are now: $\langle LC_{syn}, LC_{sem} \rangle$ and $\langle RC_{syn}, RC_{sem} \rangle$. $X(x)$ is generated as in Section 5.1, but using the new definitions of $X(x)$, LC and RC. The implementation of semantic subcat frames is similar to syntactic subcat frames. They are multisets specifying the semantic labels which the head requires in its left or right modifiers.

As an example, the probability of generating the phrase "our player 2" using NP-player $\rightarrow$

PRP$-[team](our) NN-player CD-[unum](2) is (omitting only the distance measure):

$\mathcal{P}_h$(NN-[player]|NP-[player],player)×
$\mathcal{P}_{lc}$(⟨{},{team}⟩|NP-[player],player)×
$\mathcal{P}_{rc}$(⟨{},{unum}⟩|NP-[player],player)×
$\mathcal{P}_l$(PRP$-[team](our)|NP-[player],player,⟨{},{team}⟩)×
$\mathcal{P}_r$(CD-[unum](2)|NP-[player],player,⟨{},{unum}⟩)×
$\mathcal{P}_l$(STOP|NP-[player],player,⟨{},{}⟩)×
$\mathcal{P}_r$(STOP|NP-[player],player,⟨{},{}⟩)

5.3 Smoothing

Since the left and right modifiers are independently generated in the same way, we only discuss smoothing for the left side. Each probability estimation in the above generation steps is called a *parameter*. To reduce the risk of sparse data problems, the parameters are decomposed as follows:

$$\begin{aligned}\mathcal{P}_h(H|C) &= \mathcal{P}_{h_{syn}}(H_{syn}|C) \times \\ &\quad \mathcal{P}_{h_{sem}}(H_{sem}|C, H_{syn}) \\ \mathcal{P}_{lc}(LC|C) &= \mathcal{P}_{lc_{syn}}(LC_{syn}|C) \times \\ &\quad \mathcal{P}_{lc_{sem}}(LC_{sem}|C, LC_{syn}) \\ \mathcal{P}_l(L_i(l_i)|C) &= \mathcal{P}_{l_{syn}}(L_{i_{syn}}(lt_{i_{syn}}, lw_i)|C)\times \\ &\quad \mathcal{P}_{l_{sem}}(L_{i_{sem}}(lt_{i_{sem}}, lw_i)|C, L_{i_{syn}}(lt_{i_{syn}}))\end{aligned}$$

For brevity, C is used to represent the context on which each parameter is conditioned; lw_i, $lt_{i_{syn}}$, and $lt_{i_{sem}}$ are the word, POS tag and semantic tag generated for the non-terminal L_i. The word is generated separately in the syntactic and semantic outputs.

We make the independence assumption that the syntactic output is only conditioned on syntactic features, and semantic output on semantic ones. Note that the syntactic and semantic parameters are still integrated in the model to find the globally most likely parse. The syntactic parameters are the same as in Section 5.1 and are smoothed as in (Collins, 1997). We've also tried different ways of conditioning syntactic output on semantic features and vice versa, but they didn't help. Our explanation is the integrated syntactic and semantic parameters have already captured the benefit of this integrated approach in our experimental domains.

Since the semantic parameters do not depend on any syntactic features, we omit the sem subscripts in the following discussion. As in (Collins, 1997), the parameter $\mathcal{P}_l(L_i(lt_i, lw_i)|P, H, w, t, \Delta, LC)$ is further smoothed as follows:

$$\begin{aligned}&\mathcal{P}_{l1}(L_i|P, H, w, t, \Delta, LC) \times \\ &\mathcal{P}_{l2}(lt_i|P, H, w, t, \Delta, LC, L_i) \times \\ &\mathcal{P}_{l3}(lw_i|P, H, w, t, \Delta, LC, L_i(lt_i))\end{aligned}$$

Note this smoothing is different from the syntactic counterpart. This is due to the difference between POS tags and semantic tags; namely, semantic tags are generally more specific.

Table 1 shows the various levels of back-off for each semantic parameter. The probabilities from these back-off levels are interpolated using the techniques in (Collins, 1997). All words occurring less than 3 times in the training data, and words in test data that were not seen in training, are unknown words and are replaced with the "UNKNOWN" token. Note this threshold is smaller than the one used in (Collins, 1997) since the corpora used in our experiments are smaller.

5.4 POS Tagging and Semantic Tagging

For unknown words, the POS tags allowed are limited to those seen with any unknown words during training. Otherwise they are generated along with the words using the same approach as in (Collins, 1997). When parsing, semantic tags for each known word are limited to those seen with that word during training data. The semantic tags allowed for an unknown word are limited to those seen with its associated POS tags during training.

6 Experimental Evaluation

6.1 Methodology

Two corpora of NL sentences paired with MR's were used to evaluate SCISSOR. For CLANG, 300 pieces of coaching advice were randomly selected from the log files of the 2003 RoboCup Coach Competition. Each formal instruction was translated into English by one of four annotators (Kate et al., 2005). The average length of an NL sentence in this corpus is 22.52 words. For GEOQUERY, 250 questions were collected by asking undergraduate students to generate English queries for the given database. Queries were then manually translated

BACK-OFFLEVEL	$\mathcal{P}_h(H\|...)$	$\mathcal{P}_{LC}(LC\|...)$	$\mathcal{P}_{L1}(L_i\|...)$	$\mathcal{P}_{L2}(lt_i\|...)$	$\mathcal{P}_{L3}(lw_i\|...)$
1	P,w,t	P,H,w,t	P,H,w,t,Δ,LC	P,H,w,t,Δ,LC, L_i	P,H,w,t,Δ,LC, L_i, lt_i
2	P,t	P,H,t	P,H,t,Δ,LC	P,H,t,Δ,LC, L_i	P,H,t,Δ,LC, L_i, lt_i
3	P	P,H	P,H,Δ,LC	P,H,Δ,LC, L_i	L_i, lt_i
4	–	–	–	L_i	lt_i

Table 1: Conditioning variables for each back-off level for semantic parameters (sem subscripts omitted).

into logical form (Zelle and Mooney, 1996). The average length of an NL sentence in this corpus is 6.87 words. The queries in this corpus are more complex than those in the ATIS database-query corpus used in the speech community (Zue and Glass, 2000) which makes the GEOQUERY problem harder, as also shown by the results in (Popescu et al., 2004). The average number of possible semantic tags for each word which can represent meanings in CLANG is 1.59 and that in GEOQUERY is 1.46.

SCISSOR was evaluated using standard 10-fold cross validation. NL test sentences are first parsed to generate their SAPT's, then their MR's were built from the trees. We measured the number of test sentences that produced complete MR's, and the number of these MR's that were correct. For CLANG, an MR is correct if it exactly matches the correct representation, up to reordering of the arguments of commutative operators like `and`. For GEOQUERY, an MR is correct if the resulting query retrieved the same answer as the correct representation when submitted to the database. The performance of the parser was then measured in terms of *precision* (the percentage of completed MR's that were correct) and *recall* (the percentage of all sentences whose MR's were correctly generated).

We compared SCISSOR's performance to several previous systems that learn semantic parsers that can map sentences into formal MRL's. CHILL (Zelle and Mooney, 1996) is a system based on Inductive Logic Programming (ILP). We compare to the version of CHILL presented in (Tang and Mooney, 2001), which uses the improved COCKTAIL ILP system and produces more accurate parsers than the original version presented in (Zelle and Mooney, 1996). SILT is a system that learns symbolic, pattern-based, transformation rules for mapping NL sentences to formal languages (Kate et al., 2005). It comes in two versions, SILT-string, which maps NL strings directly to an MRL, and SILT-tree, which maps syntactic

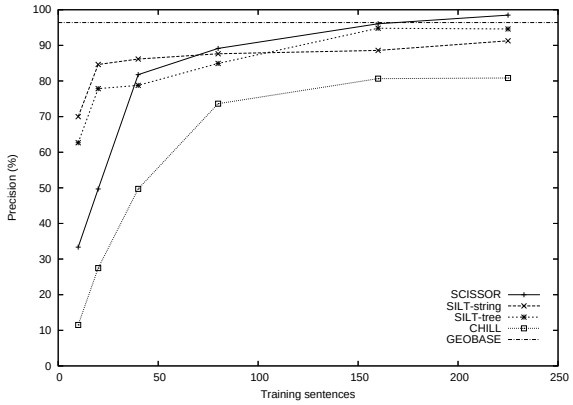


Figure 6: Precision learning curves for GEOQUERY.

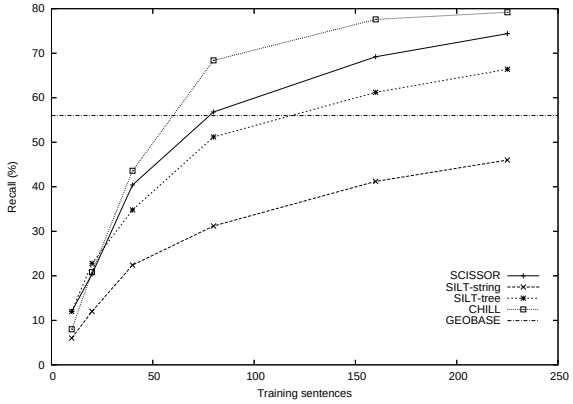


Figure 7: Recall learning curves for GEOQUERY.

parse trees (generated by the Collins parser) to an MRL. In the GEOQUERY domain, we also compare to the original hand-built parser GEOBASE.

6.2 Results

Figures 6 and 7 show the precision and recall learning curves for GEOQUERY, and Figures 8 and 9 for CLANG. Since CHILL is very memory intensive, it could not be run with larger training sets of the CLANG corpus.

Overall, SCISSOR gives the best precision and recall results in both domains. The only exception is with recall for GEOQUERY, for which CHILL is slightly higher. However, SCISSOR has significantly higher precision (see discussion in Section 7).

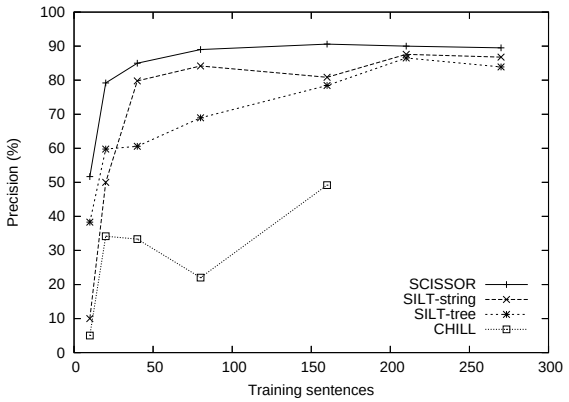


Figure 8: Precision learning curves for CLANG.

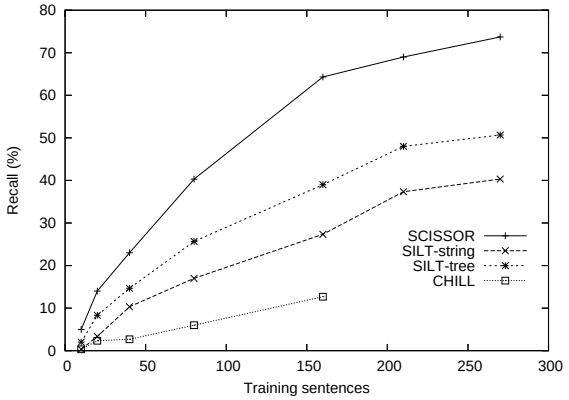


Figure 9: Recall learning curves for CLANG.

Results on a larger GEOQUERY corpus with 880 queries have been reported for PRECISE (Popescu et al., 2003): 100% precision and 77.5% recall. On the same corpus, SCISSOR obtains 91.5% precision and 72.3% recall. However, the figures are not comparable. PRECISE can return multiple distinct SQL queries when it judges a question to be ambiguous and it is considered correct when *any* of these SQL queries is correct. Our measure only considers the top result. Due to space limitations, we do not present complete learning curves for this corpus.

7 Related Work

We first discuss the systems introduced in Section 6. CHILL uses computationally-complex ILP methods, which are slow and memory intensive. The string-based version of SILT uses no syntactic information while the tree-based version generates a syntactic parse first and then transforms it into an MR. In contrast, SCISSOR integrates syntactic and semantic processing, allowing each to constrain and inform the other. It uses a successful approach to statistical parsing that attempts to find the SAPT with maximum likelihood, which improves robustness compared to purely rule-based approaches. However, SCISSOR requires an extra training input, gold-standard SAPT's, not required by these other systems. Further automating the construction of training SAPT's from sentences paired with MR's is a subject of on-going research.

PRECISE is designed to work only for the specific task of NL database interfaces. By comparison, SCISSOR is more general and can work with other MRL's as well (e.g. CLANG). Also, PRECISE is not a learning system and can fail to parse a query it considers ambiguous, even though it may not be considered ambiguous by a human and could potentially be resolved by learning regularities in the training data.

In (Lev et al., 2004), a syntax-driven approach is used to map logic puzzles described in NL to an MRL. The syntactic structures are paired with hand-written rules. A statistical parser is used to generate syntactic parse trees, and then MR's are built using compositional semantics. The meaning of open-category words (with only a few exceptions) is considered irrelevant to solving the puzzle and their meanings are not resolved. Further steps would be needed to generate MR's in other domains like CLANG and GEOQUERY. No empirical results are reported for their approach.

Several machine translation systems also attempt to generate MR's for sentences. In (et al., 2002), an English-Chinese speech translation system for limited domains is described. They train a statistical parser on trees with only semantic labels on the nodes; however, they do not integrate syntactic and semantic parsing.

History-based models of parsing were first introduced in (Black et al., 1993). Their original model also included semantic labels on parse-tree nodes, but they were not used to generate a formal MR. Also, their parsing model is impoverished compared to the history included in Collins' more recent model. SCISSOR explores incorporating semantic labels into Collins' model in order to produce a complete SAPT which is then used to generate a formal MR.

The systems introduced in (Miller et al., 1996; Miller et al., 2000) also integrate semantic labels into parsing; however, their SAPT's are used to pro-

duce a much simpler MR, i.e., a single semantic frame. A sample frame is AIRTRANSPORTATION which has three slots – the arrival time, origin and destination. Only one frame needs to be extracted from each sentence, which is an easier task than our problem in which multiple nested frames (predicates) must be extracted. The syntactic model in (Miller et al., 2000) is similar to Collins', but does not use features like subcat frames and distance measures. Also, the non-terminal label X is not further decomposed into separately-generated semantic and syntactic components. Since it used much more specific labels (the cross-product of the syntactic and semantic labels), its parameter estimates are potentially subject to much greater sparse-data problems.

8 Conclusion

SCISSOR learns statistical parsers that integrate syntax and semantics in order to produce a semantically augmented parse tree that is then used to compositionally generate a formal meaning representation. Experimental results in two domains, a natural-language database interface and an interpreter for coaching instructions in robotic soccer, have demonstrated that SCISSOR generally produces more accurate semantic representations than several previous approaches. By augmenting a state-of-the-art statistical parsing model to include semantic information, it is able to integrate syntactic and semantic clues to produce a robust interpretation that supports the generation of complete formal meaning representations.

9 Acknowledgements

We would like to thank Rohit J. Kate , Yuk Wah Wong and Gregory Kuhlmann for their help in annotating the CLANG corpus and providing the evaluation tools. This research was supported by Defense Advanced Research Projects Agency under grant HR0011-04-1-0007.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Ezra Black, Frederick Jelineck, John Lafferty, David M. Magerman, Robert L. Mercer, and Salim Roukos. 1993. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proc. of ACL-93*, pages 31–37, Columbus, Ohio.

Borland International. 1988. *Turbo Prolog 2.0 Reference Guide*. Borland International, Scotts Valley, CA.

Mao Chen et al. 2003. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

Michael J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL-97*, pages 16–23, Madrid, Spain.

Yuqing Gao et al. 2002. Mars: A statistical semantic parsing and generation-based multilingual automatic translation system. *Machine Translation*, 17:185–212.

Daniel Gildea and Daniel Jurafsky. 2002. Automated labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. To appear in *Proc. of AAAI-05*, Pittsburgh, PA.

Iddo Lev, Bill MacCartney, Christopher D. Manning, and Roger Levy. 2004. Solving logic puzzles: From robust processing to precise semantics. In *Proc. of 2nd Workshop on Text Meaning and Interpretation, ACL-04*, Barcelona, Spain.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *ACL-96*, pages 55–61, Santa Cruz, CA.

Scott Miller, Heidi Fox, Lance A. Ramshaw, and Ralph M. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc. of NAACL-00*, pages 226–233, Seattle, Washington.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proc. of IUI-2003*, pages 149–157, Miami, FL. ACM.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *COLING-04*, Geneva, Switzerland.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proc. of ECML-01*, pages 466–477, Freiburg, Germany.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proc. of AAAI-96*, pages 1050–1055, Portland, OR.

Victor W. Zue and James R. Glass. 2000. Conversational interfaces: Advances and challenges. In *Proc. of the IEEE*, volume 88(8), pages 1166–1180.

Search Engine Statistics Beyond the n-gram: Application to Noun Compound Bracketing

Preslav Nakov
EECS, Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
nakov@cs.berkeley.edu

Marti Hearst
SIMS
University of California, Berkeley
Berkeley, CA 94720
hearst@sims.berkeley.edu

Abstract

In order to achieve the long-range goal of semantic interpretation of noun compounds, it is often necessary to first determine their syntactic structure. This paper describes an unsupervised method for noun compound bracketing which extracts statistics from Web search engines using a χ^2 measure, a new set of surface features, and paraphrases. On a gold standard, the system achieves results of 89.34% (baseline 66.80%), which is a sizable improvement over the state of the art (80.70%).

1 Introduction

An important but understudied language analysis problem is that of noun compound bracketing, which is generally viewed as a necessary step towards noun compound interpretation. Consider the following contrastive pair of noun compounds:

(1) *liver cell antibody*
(2) *liver cell line*

In example (1) an *antibody* targets a *liver cell*, while (2) refers to a *cell line* which is derived from the *liver*. In order to make these semantic distinctions accurately, it can be useful to begin with the correct grouping of terms, since choosing a particular syntactic structure limits the options left for semantics. Although equivalent at the part of speech (POS) level, these two noun compounds have different syntactic trees. The distinction can be represented as a binary tree or, equivalently, as a binary bracketing:

(1b) [[*liver cell*] *antibody*] (left bracketing)
(2b) [*liver* [*cell line*]] (right bracketing)

In this paper, we describe a highly accurate unsupervised method for making bracketing decisions for noun compounds (NCs). We improve on the current standard approach of using bigram estimates to compute adjacency and dependency scores by introducing the use of the χ^2 measure for this problem. We also introduce a new set of surface features for querying Web search engines which prove highly effective. Finally, we experiment with paraphrases for improving prediction statistics. We have evaluated the application of combinations of these features to predict NC bracketing on two distinct collections, one consisting of terms drawn from encyclopedia text, and another drawn from bioscience text.

The remainder of this paper describes related work, the word association models, the surface features, the paraphrase features and the results.

2 Related Work

The syntax and semantics of NCs is an active area of research; the *Journal of Computer Speech and Language* has an upcoming special issue on Multiword Expressions.

The best known early work on automated unsupervised NC bracketing is that of Lauer (1995) who introduces the probabilistic dependency model for the syntactic disambiguation of NCs and argues against the adjacency model, proposed by Marcus (1980), Pustejovsky et al. (1993) and Resnik (1993). Lauer collects n-gram statistics from Grolier's encyclopedia, containing about 8 million words. To

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 17–24, Ann Arbor, June 2005.

overcome data sparsity problems, he estimates probabilities over conceptual categories in a taxonomy (Roget's thesaurus) rather than for individual words.

Lauer evaluated his models on a set of 244 unambiguous NCs derived from the same encyclopedia (inter-annotator agreement 81.50%) and achieved 77.50% for the dependency model above (baseline 66.80%). Adding POS and further tuning allowed him to achieve the state-of-the-art result of 80.70%.

More recently, Keller and Lapata (2003) evaluate the utility of using Web search engines for obtaining frequencies for unseen bigrams. They then later propose using Web counts as a baseline unsupervised method for many NLP tasks (Lapata and Keller, 2004). They apply this idea to six NLP tasks, including the syntactic and semantic disambiguation of NCs following Lauer (1995), and show that variations on bigram counts perform nearly as well as more elaborate methods. They do not use taxonomies and work with the word n-grams directly, achieving 78.68% with a much simpler version of the dependency model.

Girju et al. (2005) propose a *supervised* model (decision tree) for NC bracketing *in context*, based on five semantic features (requiring the correct WordNet sense to be given): the top three WordNet semantic classes for each noun, derivationally related forms and whether the noun is a nominalization. The algorithm achieves accuracy of 83.10%.

3 Models and Features

3.1 Adjacency and Dependency Models

In related work, a distinction is often made between what is called the *dependency model* and the *adjacency model*. The main idea is as follows. For a given 3-word NC $w_1w_2w_3$, there are two reasons it may take on right bracketing, $[w_1[w_2w_3]]$. Either (a) w_2w_3 is a compound (modified by w_1), or (b) w_1 and w_2 independently modify w_3. This distinction can be seen in the examples *home health care* (*health care* is a compound modified by *home*) versus *adult male rat* (*adult* and *male* independently modify *rat*).

The adjacency model checks (a), whether w_2w_3 is a compound (i.e., how strongly w_2 modifies w_3 as opposed to w_1w_2 being a compound) to decide whether or not to predict a right bracketing. The dependency model checks (b), does w_1 modify w_3 (as opposed to w_1 modifying w_2).

Left bracketing is a bit different since there is only modificational choice for a 3-word NC. If w_1 modifies w_2, this implies that w_1w_2 is a compound which in turn modifies w_3, as in *law enforcement agent*.

Thus the usefulness of the adjacency model vs. the dependency model can depend in part on the mix of left and right bracketing. Below we show that the dependency model works better than the adjaceny model, confirming other results in the literature. The next subsections describe several different ways to compute these measures.

3.2 Using Frequencies

The most straightforward way to compute adjacency and dependency scores is to simply count the corresponding frequencies. Lapata and Keller (2004) achieved their best accuracy (78.68%) with the dependency model and the simple symmetric score $\#(w_i, w_j)$.[1]

3.3 Computing Probabilities

Lauer (1995) assumes that adjacency and dependency should be computed via probabilities. Since they are relatively simple to compute, we investigate them in our experiments.

Consider the dependency model, as introduced above, and the NC $w_1w_2w_3$. Let $\Pr(w_i \rightarrow w_j|w_j)$ be the probability that the word w_i precedes a given fixed word w_j. Assuming that the distinct head-modifier relations are independent, we obtain $\Pr(\text{right}) = \Pr(w_1 \rightarrow w_3|w_3)\Pr(w_2 \rightarrow w_3|w_3)$ and $\Pr(\text{left}) = \Pr(w_1 \rightarrow w_2|w_2)\Pr(w_2 \rightarrow w_3|w_3)$. To choose the more likely structure, we can drop the shared factor and compare $\Pr(w_1 \rightarrow w_3|w_3)$ to $\Pr(w_1 \rightarrow w_2|w_2)$.

The alternative adjacency model compares $\Pr(w_2 \rightarrow w_3|w_3)$ to $\Pr(w_1 \rightarrow w_2|w_2)$, i.e. the association strength between the last two words vs. that between the first two. If the first probability is larger than the second, the model predicts right.

The probability $\Pr(w_1 \rightarrow w_2|w_2)$ can be estimated as $\#(w_1, w_2)/\#(w_2)$, where $\#(w_1, w_2)$ and $\#(w_2)$ are the corresponding bigram and unigram

[1] This score worked best on training, when Keller&Lapata were doing model selection. On testing, $\Pr$ (with the dependency model) worked better and achieved accuracy of 80.32%, but this result was ignored, as $\Pr$ did worse on training.

frequencies. They can be approximated as the number of pages returned by a search engine in response to queries for the exact phrase "$w_1\ w_2$" and for the word w_2. In our experiments below we smoothed[2] each of these frequencies by adding 0.5 to avoid problems caused by nonexistent n-grams.

Unless some particular probabilistic interpretation is needed,[3] there is no reason why for a given ordered pair of words (w_i, w_j), we should use $\Pr(w_i \rightarrow w_j|w_j)$ rather than $\Pr(w_j \rightarrow w_i|w_i)$, $i < j$. This is confirmed by the adjacency model experiments in (Lapata and Keller, 2004) on Lauer's NC set. Their results show that both ways of computing the probabilities make sense: using Altavista queries, the former achieves a higher accuracy (70.49% vs. 68.85%), but the latter is better on the British National Corpus (65.57% vs. 63.11%).

3.4 Other Measures of Association

In both models, the probability $\Pr(w_i \rightarrow w_j|w_j)$ can be replaced by some (possibly symmetric) measure of association between w_i and w_j, such as *Chi squared* (χ^2). To calculate $\chi^2(w_i, w_j)$, we need:

- (A) $\#(w_i, w_j)$;
- (B) $\#(w_i, \overline{w_j})$, the number of bigrams in which the first word is w_i, followed by a word other than w_j;
- (C) $\#(\overline{w_i}, w_j)$, the number of bigrams, ending in w_j, whose first word is other than w_i;
- (D) $\#(\overline{w_i}, \overline{w_j})$, the number of bigrams in which the first word is not w_i and the second is not w_j.

They are combined in the following formula:

$$\chi^2 = \frac{N(AD - BC)^2}{(A + C)(B + D)(A + B)(C + D)} \quad (1)$$

Here $N = A + B + C + D$ is the total number of bigrams, $B = \#(w_i) - \#(w_i, w_j)$ and $C = \#(w_j) - \#(w_i, w_j)$. While it is hard to estimate D directly, we can calculate it as $D = N - A - B - C$. Finally, we estimate N as the total number of indexed bigrams on the Web. They are estimated as 8 trillion, since Google indexes about 8 billion pages and each contains about 1,000 words on average.

Other measures of word association are possible, such as *mutual information* (MI), which we can use with the dependency and the adjacency models, similarly to #, χ^2 or Pr. However, in our experiments, χ^2 worked better than other methods; this is not surprising, as χ^2 is known to outperform MI as a measure of association (Yang and Pedersen, 1997).

3.5 Web-Derived Surface Features

Authors sometimes (consciously or not) disambiguate the words they write by using surface-level markers to suggest the correct meaning. We have found that exploiting these markers, when they occur, can prove to be very helpful for making bracketing predictions. The enormous size of Web search engine indexes facilitates finding such markers frequently enough to make them useful.

One very productive feature is the *dash* (hyphen). Starting with the term *cell cycle analysis*, if we can find a version of it in which a dash occurs between the first two words: *cell-cycle*, this suggests a left bracketing for the full NC. Similarly, the dash in *donor T-cell* favors a right bracketing. The right-hand dashes are less reliable though, as their scope is ambiguous. In *fiber optics-system*, the hyphen indicates that the noun compound *fiber optics* modifies *system*. There are also cases with multiple hyphens, as in *t-cell-depletion*, which preclude their use.

The genitive ending, or *possessive* marker is another useful indicator. The phrase *brain's stem cells* suggests a right bracketing for *brain stem cells*, while *brain stem's cells* favors a left bracketing.[4]

Another highly reliable source is related to internal *capitalization*. For example *Plasmodium vivax Malaria* suggests left bracketing, while *brain Stem cells* would favor a right one. (We disable this feature on Roman digits and single-letter words to prevent problems with terms like *vitamin D deficiency*, where the capitalization is just a convention as opposed to a special mark to make the reader think that the last two terms should go together.)

[2] Zero counts sometimes happen for $\#(w_1, w_3)$, but are rare for unigrams and bigrams on the Web, and there is no need for a more sophisticated smoothing.

[3] For example, as used by Lauer to introduce a prior for left-right bracketing preference. The best Lauer model does not work with words directly, but uses a taxonomy and further needs a probabilistic interpretation so that the hidden taxonomy variables can be summed out. Because of that summation, the term $\Pr(w_2 \rightarrow w_3|w_3)$ does not cancel in his dependency model.

[4] Features can also occur combined, e.g. *brain's stem-cells*.

We can also make use of embedded *slashes*. For example in *leukemia/lymphoma cell*, the slash predicts a right bracketing since the first word is an alternative and cannot be a modifier of the second one.

In some cases we can find instances of the NC in which one or more words are enclosed in parentheses, e.g., *growth factor (beta)* or *(growth factor) beta*, both of which indicate a left structure, or *(brain) stem cells*, which suggests a right bracketing.

Even a comma, a dot or a colon (or any special character) can act as indicators. For example, "*health care, provider*" or "*lung cancer: patients*" are weak predictors of a left bracketing, showing that the author chose to keep two of the words together, separating out the third one.

We can also exploit dashes to words external to the target NC, as in $mouse$*-brain stem cells*, which is a weak indicator of right bracketing.

Unfortunately, Web search engines ignore punctuation characters, thus preventing querying directly for terms containing hyphens, brackets, apostrophes, etc. We collect them indirectly by issuing queries with the NC as an exact phrase and then post-processing the resulting summaries, looking for the surface features of interest. Search engines typically allow the user to explore up to 1000 results. We collect all results and summary texts that are available for the target NC and then search for the surface patterns using regular expressions over the text. Each match increases the score for left or right bracketing, depending on which the pattern favors.

While some of the above features are clearly more reliable than others, we do not try to weight them. For a given NC, we post-process the returned Web summaries, then we find the number of left-predicting surface feature instances (regardless of their type) and compare it to the number of right-predicting ones to make a bracketing decision.[5]

3.6 Other Web-Derived Features

Some features can be obtained by using the overall counts returned by the search engine. As these counts are derived from the entire Web, as opposed to a set of up to 1,000 summaries, they are of different magnitude, and we did not want to simply add them to the surface features above. They appear as independent models in Tables 1 and 2.

First, in some cases, we can query for *possessive markers* directly: although search engines drop the apostrophe, they keep the *s*, so we can query for *"brain's"* (but not for *"brains' "*). We then compare the number of times the possessive marker appeared on the second vs. the first word, to make a bracketing decision.

Abbreviations are another important feature. For example, *"tumor necrosis factor (NF)"* suggests a right bracketing, while *"tumor necrosis (TN) factor"* would favor left. We would like to issue exact phrase queries for the two patterns and see which one is more frequent. Unfortunately, the search engines drop the brackets and ignore the capitalization, so we issue queries with the parentheses removed, as in *"tumor necrosis factor nf"*. This produces highly accurate results, although errors occur when the abbreviation is an existing word (e.g., *me*), a Roman digit (e.g., *IV*), a state (e.g., *CA*), etc.

Another reliable feature is *concatenation*. Consider the NC *health care reform*, which is left-bracketed. Now, consider the bigram *"health care"*. At the time of writing, Google estimates 80,900,000 pages for it as an exact term. Now, if we try the word *healthcare* we get 80,500,000 hits. At the same time, *carereform* returns just 109. This suggests that authors sometimes concatenate words that act as compounds. We find below that comparing the frequency of the concatenation of the left bigram to that of the right (adjacency model for concatenations) often yields accurate results. We also tried the dependency model for concatenations, as well as the concatenations of two words in the context of the third one (i.e., compare frequencies of *"healthcare reform"* and *"health carereform"*).

We also used Google's support for "*", which allows a single word wildcard, to see how often two of the words are present but separated from the third by some other word(s). This implicitly tries to capture paraphrases involving the two sub-concepts making up the whole. For example, we compared the frequency of *"health care * reform"* to that of *"health * care reform"*. We also used 2 and 3 stars and switched the word group order (indicated with *rev.* in Tables 1 and 2), e.g., *"care reform * * health"*. We also tried a simple *reorder* without inserting stars, i.e., compare the frequency of *"reform health*

[5] This appears as *Surface features (sum)* in Tables 1 and 2.

care" to that of "*care reform health*". For example, when analyzing *myosin heavy chain* we see that *heavy chain myosin* is very frequent, which provides evidence against grouping *heavy* and *chain* together as they can commute.

Further, we tried to look inside the *internal inflection variability*. The idea is that if *"tyrosine kinase activation"* is left-bracketed, then the first two words probably make a whole and thus the second word can be found inflected elsewhere but the first word cannot, e.g., *"tyrosine kinases activation"*. Alternatively, if we find different internal inflections of the first word, this would favor a right bracketing.

Finally, we tried switching the word order of the first two words. If they independently modify the third one (which implies a right bracketing), then we could expect to see also a form with the first two words switched, e.g., if we are given *"adult male rat"*, we would also expect *"male adult rat"*.

3.7 Paraphrases

Warren (1978) proposes that the semantics of the relations between words in a noun compound are often made overt by paraphrase. As an example of *prepositional paraphrase*, an author describing the concept of *brain stem cells* may choose to write it in a more expanded manner, such as *stem cells in the brain*. This contrast can be helpful for syntactic bracketing, suggesting that the full NC takes on right bracketing, since *stem* and *cells* are kept together in the expanded version. However, this NC is ambiguous, and can also be paraphrased as *cells from the brain stem*, implying a left bracketing.

Some NCs' meaning cannot be readily expressed with a prepositional paraphrase (Warren, 1978). An alternative is the *copula paraphrase*, as in *office building that/which is a skyscraper* (right bracketing), or a *verbal paraphrase* such as *pain associated with arthritis migraine* (left).

Other researchers have used prepositional paraphrases as a proxy for determining the semantic relations that hold between nouns in a compound (Lauer, 1995; Keller and Lapata, 2003; Girju et al., 2005). Since most NCs have a prepositional paraphrase, Lauer builds a model trying to choose between the most likely candidate prepositions: *of*, *for*, *in*, *at*, *on*, *from*, *with* and *about* (excluding *like* which is mentioned by Warren). This could be problematic though, since as a study by Downing (1977) shows, when no context is provided, people often come up with incompatible interpretations.

In contrast, we use paraphrases in order to make syntactic bracketing assignments. Instead of trying to manually decide the correct paraphrases, we can issue queries using paraphrase patterns and find out how often each occurs in the corpus. We then add up the number of hits predicting a left versus a right bracketing and compare the counts.

Unfortunately, search engines lack linguistic annotations, making general verbal paraphrases too expensive. Instead we used a small set of hand-chosen paraphrases: *associated with*, *caused by*, *contained in*, *derived from*, *focusing on*, *found in*, *involved in*, *located at/in*, *made of*, *performed by*, *preventing*, *related to* and *used by/in/for*. It is however feasible to generate queries predicting left/right bracketing with/without a determiner for every preposition.[6] For the copula paraphrases we combine two verb forms *is* and *was*, and three complementizers *that*, *which* and *who*. These are optionally combined with a preposition or a verb form, e.g. *themes that are used in science fiction*.

4 Evaluation

4.1 Lauer's Dataset

We experimented with the dataset from (Lauer, 1995), in order to produce results comparable to those of Lauer and Keller & Lapata. The set consists of 244 unambiguous 3-noun NCs extracted from *Grolier's encyclopedia*; however, only 216 of these NCs are unique.

Lauer (1995) derived n-gram frequencies from the *Grolier's* corpus and tested the dependency and the adjacency models using this text. To help combat data sparseness issues he also incorporated a taxonomy and some additional information (see Related Work section above). Lapata and Keller (2004) derived their statistics from the Web and achieved results close to Lauer's using simple lexical models.

4.2 Biomedical Dataset

We constructed a new set of noun compounds from the biomedical literature. Using the Open NLP

[6] In addition to the articles (*a*, *an*, *the*), we also used quantifiers (e.g. *some*, *every*) and pronouns (e.g. *this*, *his*).

tools,[7] we sentence splitted, tokenized, POS tagged and shallow parsed a set of 1.4 million MEDLINE abstracts (citations between 1994 and 2003). Then we extracted all 3-noun sequences falling in the last three positions of noun phrases (NPs) found in the shallow parse. If the NP contained other nouns, the sequence was discarded. This allows for NCs which are modified by adjectives, determiners, and so on, but prevents extracting 3-noun NCs that are part of longer NCs. For details, see (Nakov et al., 2005).

This procedure resulted in 418,678 different NC types. We manually investigated the most frequent ones, removing those that had errors in tokenization (e.g., containing words like *transplan* or *tation*), POS tagging (e.g., *acute lung injury*, where *acute* was wrongly tagged as a noun) or shallow parsing (e.g., *situ hybridization*, that misses *in*). We had to consider the first 843 examples in order to obtain 500 good ones, which suggests an extraction accuracy of 59%. This number is low mainly because the tokenizer handles dash-connected words as a single token (e.g. *factor-alpha*) and many tokens contained other special characters (e.g. *cd4+*), which cannot be used in a query against a search engine and had to be discarded.

The 500 NCs were annotated independently by two judges, one of which has a biomedical background; the other one was one of the authors. The problematic cases were reconsidered by the two judges and after agreement was reached, the set contained: 361 left bracketed, 69 right bracketed and 70 ambiguous NCs. The latter group was excluded from the experiments.[8]

We calculated the inter-annotator agreement on the 430 cases that were marked as unambiguous after agreement. Using the original annotator's choices, we obtained an agreement of 88% or 82%, depending on whether we consider the annotations, that were initially marked as ambiguous by one of the judges to be correct. The corresponding values for the kappa statistics were .606 (substantial agreement) and .442 (moderate agreement).

[7] http://opennlp.sourceforge.net/

[8] Two NCs can appear more than once but with a different inflection or with a different word variant, e.g,. *colon cancer cells* and *colon carcinoma cells*.

4.3 Experiments

The n-grams, surface features, and paraphrase counts were collected by issuing exact phrase queries, limiting the pages to English and requesting filtering of similar results.[9] For each NC, we generated all possible word inflections (e.g., *tumor* and *tumors*) and alternative word variants (e.g., *tumor* and *tumour*). For the biomedical dataset they were automatically obtained from the UMLS Specialist lexicon.[10] For Lauer's set we used Carroll's morphological tools.[11] For bigrams, we inflect only the second word. Similarly, for a prepositional paraphrase we generate all possible inflected forms for the two parts, before and after the preposition.

4.4 Results and Discussion

The results are shown in Tables 1 and 2. As NCs are left-bracketed at least 2/3rds of the time (Lauer, 1995), a straightforward baseline is to always assign a left bracketing. Tables 1 and 2 suggest that the surface features perform best. The paraphrases are equally good on the biomedical dataset, but on Lauer's set their performance is lower and is comparable to that of the dependency model.

The dependency model clearly outperforms the adjacency one (as other researchers have found) on Lauer's set, but not on the biomedical set, where it is equally good. χ^2 barely outperforms #, but on the biomedical set χ^2 is a clear winner (by about 1.5%) on both dependency and adjacency models.

The frequencies (#) outperform or at least rival the probabilities on both sets and for both models. This is not surprising, given the previous results by Lapata and Keller (2004). Frequencies also outperform Pr on the biomedical set. This may be due to the abundance of single-letter words in that set (because of terms like *T cell*, *B cell*, *vitamin D* etc.; similar problems are caused by Roman digits like *ii*, *iii* etc.), whose Web frequencies are rather unreliable, as they are used by Pr but not by frequencies. Single-letter words cause potential problems for the paraphrases

[9] In our experiments we used MSN Search statistics for the n-grams and the paraphrases (unless the pattern contained a "*"), and Google for the surface features. MSN always returned exact numbers, while Google and Yahoo rounded their page hits, which generally leads to lower accuracy (Yahoo was better than Google for these estimates).

[10] http://www.nlm.nih.gov/pubs/factsheets/umlslex.html

[11] http://www.cogs.susx.ac.uk/lab/nlp/carroll/morph.html

Model	√	×	∅	P(%)	C(%)
# adjacency	183	61	0	75.00	100.00
Pr adjacency	180	64	0	73.77	100.00
MI adjacency	182	62	0	74.59	100.00
χ^2 adjacency	184	60	0	**75.41**	100.00
# dependency	193	50	1	79.42	99.59
Pr dependency	194	50	0	79.51	100.00
MI dependency	194	50	0	79.51	100.00
χ^2 dependency	195	50	0	**79.92**	100.00
# adjacency (*)	152	41	51	78.76	79.10
# adjacency (**)	162	43	39	79.02	84.02
# adjacency (***)	150	51	43	74.63	82.38
# adjacency (*, rev.)	163	48	33	77.25	86.47
# adjacency (**, rev.)	165	51	28	76.39	88.52
# adjacency (***, rev.)	156	57	31	73.24	87.30
Concatenation adj.	175	48	21	78.48	91.39
Concatenation dep.	167	41	36	**80.29**	85.25
Concatenation triples	76	3	165	**96.20**	32.38
Inflection Variability	69	36	139	65.71	43.03
Swap first two words	66	38	140	63.46	42.62
Reorder	112	40	92	73.68	62.30
Abbreviations	21	3	220	**87.50**	9.84
Possessives	32	4	208	**88.89**	14.75
Paraphrases	174	38	32	**82.08**	86.89
Surface features (sum)	183	31	30	**85.51**	87.70
Majority vote	210	22	12	90.52	95.08
Majority vote → left	*218*	*26*	*0*	***89.34***	***100.00***
Baseline (choose left)	163	81	0	66.80	100.00

Table 1: **Lauer Set.** Shown are the numbers for correct (√), incorrect (×), and no prediction (∅), followed by precision (P, calculated over √ and × only) and coverage (C, % examples with prediction). We use "→" for back-off to another model in case of ∅.

Model	√	×	∅	P(%)	C(%)
# adjacency	374	56	0	86.98	100.00
Pr adjacency	353	77	0	82.09	100.00
MI adjacency	372	58	0	86.51	100.00
χ^2 adjacency	379	51	0	**88.14**	100.00
# dependency	374	56	0	86.98	100.00
Pr dependency	369	61	0	85.81	100.00
MI dependency	369	61	0	85.81	100.00
χ^2 dependency	380	50	0	**88.37**	100.00
# adjacency (*)	373	57	0	86.74	100.00
# adjacency (**)	358	72	0	83.26	100.00
# adjacency (***)	334	88	8	79.15	98.14
# adjacency (*, rev.)	370	59	1	86.25	99.77
# adjacency (**, rev.)	367	62	1	85.55	99.77
# adjacency (***, rev.)	351	79	0	81.63	100.00
Concatenation adj.	370	47	13	88.73	96.98
Concatenation dep.	366	43	21	**89.49**	95.12
Concatenation triple	238	37	155	**86.55**	63.95
Inflection Variability	198	49	183	80.16	57.44
Swap first two words	90	18	322	83.33	25.12
Reorder	320	78	32	80.40	92.56
Abbreviations	133	23	274	**85.25**	36.27
Possessives	48	7	375	**87.27**	12.79
Paraphrases	383	44	3	**89.70**	99.30
Surface features (sum)	382	48	0	**88.84**	100.00
Majority vote	403	17	10	95.95	97.67
Majority vote → right	*410*	*20*	*0*	***95.35***	***100.00***
Baseline (choose left)	361	69	0	83.95	100.00

Table 2: **Biomedical Set.**

as well, by returning too many false positives, but they work very well with concatenations and dashes: e.g., *T cell* is often written as *Tcell*.

As Table 4 shows, most of the surface features that we predicted to be right-bracketing actually indicated left. Overall, the surface features were very good at predicting left bracketing, but unreliable for right-bracketed examples. This is probably in part due to the fact that they look for adjacent words, i.e., they act as a kind of adjacency model.

We obtained our best overall results by combining the most reliable models, marked in bold in Tables 1, 2 and 4. As they have independent errors, we used a majority vote combination.

Table 3 compares our results to those of Lauer (1995) and of Lapata and Keller (2004). It is important to note though, that our results are *directly* comparable to those of Lauer, while the Keller&Lapata's are not, since they used half of the Lauer set for development and the other half for testing.[12] We, following Lauer, used everything for testing. Lapata & Keller also used the AltaVista search engine, which no longer exists in its earlier form. The table does not contain the results of Girju et al. (2005), who achieved 83.10% accuracy, but used a *supervised* algorithm and targeted bracketing *in context*. They further "shuffled" the Lauer's set, mixing it with additional data, thus making their results even harder to compare to these in the table.

Note that using page hits as a proxy for n-gram frequencies can produce some counter-intuitive results. Consider the bigrams w_1w_4, w_2w_4 and w_3w_4 and a page that contains each bigram exactly once. A search engine will contribute a page count of 1 for w_4 instead of a frequency of 3; thus the page hits for w_4 can be smaller than the page hits for the sum of the individual bigrams. See Keller and Lapata (2003) for more issues.

[12] In fact, the differences are negligible; their system achieves pretty much the same result on the half split as well as on the whole set (personal communication).

Model	Acc. %
LEFT (baseline)	66.80
Lauer adjacency	68.90
Lauer dependency	77.50
Our χ^2 dependency	*79.92*
Lauer tuned	80.70
"Upper bound" (humans - Lauer)	81.50
Our majority vote → left	***89.34***
Keller&Lapata: LEFT (baseline)	63.93
Keller&Lapata: best BNC	68.03
Keller&Lapata: best AltaVista	78.68

Table 3: **Comparison to previous unsupervised results on *Lauer's set.*** The results of Keller & Lapata are on half of Lauer's set and thus are only indirectly comparable (note the different baseline).

5 Conclusions and Future Work

We have extended and improved upon the state-of-the-art approaches to NC bracketing using an unsupervised method that is more robust than Lauer (1995) and more accurate than Lapata and Keller (2004). Future work will include testing on NCs consisting of more than 3 nouns, recognizing the ambiguous cases, and bracketing NPs that include determiners and modifiers. We plan to test this approach on other important NLP problems.

As mentioned above, NC bracketing should be helpful for semantic interpretation. Another possible application is the refinement of parser output. Currently, NPs in the Penn TreeBank are flat, without internal structure. Absent any other information, probabilistic parsers typically assume right bracketing, which is incorrect about 2/3rds of the time for 3-noun NCs. It may be useful to augment the Penn TreeBank with dependencies inside the currently flat NPs, which may improve their performance overall.

Acknowledgements We thank Dan Klein, Frank Keller and Mirella Lapata for valuable comments, Janice Hamer for the annotations, and Mark Lauer for his dataset. This research was supported by NSF DBI-0317510, and a gift from Genentech.

Example	Predicts	Accuracy	Coverage
brain-stem cells	left	**88.22**	**92.79**
brain stem's cells	left	**91.43**	**16.28**
(brain stem) cells	left	96.55	6.74
brain stem (cells)	left	100.00	1.63
brain stem, cells	left	96.13	42.09
brain stem: cells	left	97.53	18.84
brain stem cells-death	left	80.69	60.23
brain stem cells/tissues	left	83.59	45.35
brain stem Cells	left	**90.32**	**36.04**
brain stem/cells	left	100.00	7.21
brain. stem cells	left	97.58	38.37
brain stem-cells	*right*	***25.35***	***50.47***
brain's stem cells	right	**55.88**	**7.90**
(brain) stem cells	*right*	*46.67*	*3.49*
brain (stem cells)	*right*	*0.00*	*0.23*
brain, stem cells	right	54.84	14.42
brain: stem cells	*right*	*44.44*	*6.28*
rat-brain stem cells	*right*	*17.97*	*68.60*
neural/brain stem cells	*right*	*16.36*	*51.16*
brain Stem cells	*right*	*24.69*	*18.84*
brain/stem cells	right	53.33	3.49
brain stem. cells	*right*	*39.34*	*14.19*

Table 4: **Surface features analysis (%s)**, run over the biomedical set.

References

Pamela Downing. 1977. On the creation and use of english compound nouns. *Language*, (53):810–842.

R. Girju, D. Moldovan, M. Tatu, and D. Antohe. 2005. On the semantics of noun compounds. *Journal of Computer Speech and Language - Special Issue on Multiword Expressions*.

Frank Keller and Mirella Lapata. 2003. Using the Web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29:459–484.

Mirella Lapata and Frank Keller. 2004. The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. In *Proceedings of HLT-NAACL*, pages 121–128, Boston.

Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. Ph.D. thesis, Department of Computing Macquarie University NSW 2109 Australia.

Mitchell Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. MIT Press.

Preslav Nakov, Ariel Schwartz, Brian Wolf, and Marti Hearst. 2005. Scaling up BioNLP: Application of a text annotation architecture to noun compound bracketing. In *Proceedings of SIG BioLINK*.

James Pustejovsky, Peter Anick, and Sabine Bergler. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

Philip Resnik. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania, UMI Order No. GAX94-13894.

Beatrice Warren. 1978. Semantic patterns of noun-noun compounds. In *Gothenburg Studies in English 41, Goteburg, Acta Universtatis Gothoburgensis*.

Y. Yang and J. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of ICML'97)*, pages 412–420.

New Experiments in Distributional Representations of Synonymy

Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow,
Sadik Kapadia, Richard Rohwer, Zhiqiang Wang
HNC Software, LLC
3661 Valley Centre Drive
San Diego, CA 92130, USA
{DayneFreitag,MatthiasBlume,JohnByrnes,EdChow,
SadikKapadia,RichardRohwer,ZhiqiangWang}@fairisaac.com

Abstract

Recent work on the problem of detecting synonymy through corpus analysis has used the Test of English as a Foreign Language (TOEFL) as a benchmark. However, this test involves as few as 80 questions, prompting questions regarding the statistical significance of reported results. We overcome this limitation by generating a TOEFL-like test using WordNet, containing thousands of questions and composed only of words occurring with sufficient corpus frequency to support sound distributional comparisons. Experiments with this test lead us to a similarity measure which significantly outperforms the best proposed to date. Analysis suggests that a strength of this measure is its relative robustness against polysemy.

1 Introduction

Many text applications are predicated on the idea that shallow lexical semantics can be acquired through corpus analysis. Harris articulated the expectation that words with similar meanings would be used in similar contexts (Harris, 1968), and recent empirical work involving large corpora has borne this out. In particular, by associating each word with a distribution over the words observed in its context, we can distinguish synonyms from non-synonyms with fair reliability. This capability may be exploited to generate corpus-based thesauri automatically (Lin, 1998), or used in any other application of text that might benefit from a measure of lexical semantic similarity. And synonymy is a logical first step in a broader research program that seeks to account for natural language semantics through distributional means.

Previous research into corpus-analytic approaches to synonymy has used the Test of English as a Foreign Language (*TOEFL*). The TOEFL consists of 300 multiple-choice question, each question involving five words: the problem or target word and four response words, one of which is a synonym of the target. The objective is to identify the synonym (call this the *answer* word, and call the other response words *decoys*). In the context of research into lexical semantics, we seek a distance function which as reliably as possible orders the answer word in front of the decoys.

Landauer and Dumais first proposed the TOEFL as a test of lexical semantic similarity and reported a score of 64.4% on an 80-question version of the TOEFL, a score nearly identical to the average score of human test takers (Landauer and Dumais, 1997). Subsequently, Sahlgren reported a score of 72.0% on the same test using "random indexing" and a different training corpus (Sahlgren, 2001). By analyzing a much larger corpus, Ehlert was able to score 82% on a 300-question version of the TOEFL, using a simple distribution over contextual words (Ehlert, 2003).

While success on the TOEFL does not immediately guarantee success in real-word applications requiring lexical similarity judgments, the scores have an intuitive appeal. They are easily interpretable, and the expected performance of a random

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 25–32, Ann Arbor, June 2005.

guesser (25%) and typical human performance are both known. Nevertheless, the TOEFL is problematic in at least two ways. On the one hand, because it involves so few questions, conclusions based on the TOEFL regarding closely competing approaches are suspect. Even on the 300-question TOEFL, a score of 82% is accurate only to within plus or minus 4% at the 95% confidence level. The other shortcoming is a potential mis-match between the test vocabulary and the corpus vocabulary. Typically, a substantial number of questions include words observed too infrequently in the training corpus for a semantic judgment to be made with any confidence.

We seek to overcome these difficulties by generating TOEFL-like tests automatically from WordNet (Fellbaum, 1998). While WordNet has been used before to evaluate corpus-analytic approaches to lexical similarity (Lin, 1998), the metric proposed in that study, while useful for comparative purposes, lacks an intuitive interpretation. In contrast, we emulate the TOEFL using WordNet and inherit the TOEFL's easy interpretability.

Given a corpus, we first derive a list of words occurring with sufficient marginal frequency to support a distributional comparison. We then use WordNet to generate a large set of questions identical in format to those in the TOEFL. For a vocabulary of reasonable size, this yields questions numbering in the thousands. While the resulting questions differ in some interesting ways from those in the TOEFL (see below), their sheer number supports more confident conclusions. Beyond this, we can partition them by part of speech or degree of polysemy, enabling some analyses not supported by the original TOEFL.

2 The Test

To generate a TOEFL-like test from WordNet, we perform the following procedure once each for nouns, verbs, adjectives and adverbs. Given a list of candidate words, we produce one test question for every ordered pair of words appearing together in any synset in the respective WordNet part-of-speech database. Decoy words are chosen at random from among other words in the database that do not have a synonymy relation with either word in the pair. For convenience, we will call the resulting test the *WordNet-based synonymy test* (WBST).

```
technology:
  A. engineering  B. difference
  C. department   D. west
stadium:
  A. miss         B. hockey
  C. wife         D. bowl
string:
  A. giant        B. ballet
  C. chain        D. hat
trial:
  A. run          B. one-third
  C. drove        D. form
```

Table 1: Four questions chosen at random from the noun test. Answers are A, D, C, and A.

We take a few additional steps in order to increase the resemblance between the WBST and the TOEFL. First, we remove from consideration any stop words or inflected forms. Note that whether a particular wordform is inflected is a function of its presumed part of speech. The word "indicted" is either an inflected verb (so would not be used as a word in a question involving verbs) or an uninflected adjective. Second, we rule out pairs of words that are too similar under the string edit distance. Morphological variants often share a synset in WordNet. For example, "group" and "grouping" share a nominal sense. Questions using such pairs appear trivial to human test takers and allow stemming shortcuts.

In the experiments reported in this paper, we used WordNet 1.7.1. Our experimental corpus is the North American News corpus, which is also used by Ehlert (2003). We include as a candidate test word any word occurring at least 1000 times in the corpus (about 15,000 words when restricted to those appearing in WordNet). Table 1 shows four sample questions generated from this list out of the noun database. In total, this procedure yields 9887 noun, 7398 verb, 5824 adjective, and 461 adverb questions, a total of 23,570 questions.[1]

This procedure yields questions that differ in some interesting ways from those in the TOEFL. Most notable is a bias in favor of polysemous terms. The number of times a word appears as either the target or the answer word is proportional to the number of synonyms it has in the candidate list. In contrast,

[1]This test is available as `http://www.cs.cmu.edu/~dayne/wbst-nanews.tar.gz`.

decoy words are chosen at random, so are less polysemous on average.

3 The Space of Solutions

Given that we have a large number of test questions composed of words with high corpus frequencies, we now seek to optimize performance on the WBST. The solutions we consider all start with a word-conditional context frequency vector, usually normalized to form a probability distribution. We answer a question by comparing the target term vector and each of the response term vectors, choosing the "closest."

This problem definition excludes a common class of solutions to this problem, in which the closeness of a pair of terms is a statistic of the co-occurrence patterns of the specific terms in question. It has been shown that measures based on the pointwise mutual information (PMI) between question words yield good results on the TOEFL (Turney, 2001; Terra and Clarke, 2003). However, Ehlert (2003) shows convincingly that, for a fixed amount of data, the distributional model performs better than what we might call the pointwise co-occurrence model. Terra and Clark (2003) report a top score of 81.3% on an 80-word version of the TOEFL, which compares favorably with Ehlert's best of 82% on a 300-word version, but their corpus is approximately 200 times as large as Ehlert's.

Note that these two approaches are complementary and can be combined in a supervised setting, along with static resources, to yield truly strong performance (97.5%) on the TOEFL (Turney et al., 2003). While impressive, this work begs an important question: Where do we obtain the training data when moving to a less commonly taught language, to say nothing of the comprehensive thesauri and Web resources? In this paper, we focus on shallow methods that use only the text corpus. We are interested less in optimizing performance on the TOEFL than in investigating the validity and limits of the distributional hypothesis, and in illuminating the barriers to automated human-level lexical similarity judgments.

3.1 Definitions of Context

As in previous work, we form our context distributions by recording word-conditional counts of feature occurrences within some fixed window of a reference token. In this study, features are just unnormalized tokens, possibly augmented with direction and distance information. In other words, we do not investigate the utility of stemming. Similarly, except where noted, we do not remove stop words.

All context definitions involve a window size, which specifies the number of tokens to consider on *either* side of an occurrence of a reference term. It is always symmetric. Thus, a window size of one indicates that only the immediately adjacent tokens on either side should be considered. By default, we bracket a token sequence with pseudo-tokens "<bos>" and "<eos>".[2]

Contextual tokens in the window may be either observed or disregarded, and the policy governing which to admit is one of the dimensions we explore here. The decision whether or not to observe a particular contextual token is made before counting commences, and is not sensitive to the circumstances of a particular occurrence (e.g., its participation in some syntactic relation (Lin, 1997; Lee, 1999)). When a contextual token is observed, it is always counted as a single occurrence. Thus, in contrast with earlier approaches (Sahlgren, 2001; Ehlert, 2003), we do not use a weighting scheme that is a function of distance from the reference token.

Once we have chosen to observe a contextual token, additional parameters govern whether counting should be sensitive to the side of the reference token on which it occurs and how distant from the reference token it is. If the *strict direction* parameter is true, a left occurrence is distinguished from a right occurrence. If *strict distance* is true, occurrences at distinct removes (in number of tokens) are recorded as distinct event types.

3.2 Distance Measures

The product of a particular context policy is a co-occurrence matrix N, where the contents of a cell $N_{w,c}$ is the number of times context c is observed to occur with word w. A row of this matrix (N_w) is

[2] In this paper, a sequence is a North American News segment delimited by the <p> tag. Nominally paragraphs, most of these segments are single sentences.

therefore a word-conditional context frequency vector. In comparing two of these vectors, we typically normalize counts so that all cells in a row sum to one, yielding a word-conditional distribution over contexts $P(c|w)$ (but see the Cosine measure below).

We investigate some of the distance measures commonly employed in comparing term vectors. These include:

Manhattan	$\sum_i \lvert P(c_i\lvert w_1) - P(c_i\lvert w_2)\rvert$
Euclidean	$\sqrt{\sum_i [P(c_i\lvert w_1) - P(c_i\lvert w_2)]^2}$
Cosine	$\frac{\sum_i N_{w_1,c_i} N_{w_2,c_i}}{\lvert\lvert N_{w_1}\rvert\rvert \cdot \lvert\lvert N_{w_2}\rvert\rvert}$

Note that whereas we use probabilities in calculating the Manhattan and Euclidean distances, in order to avoid magnitude effects, the Cosine, which defines a different kind of normalization, is applied to raw number counts.

We also avail ourselves of measures suggested by probability theory. For $\delta \in (0,1)$ and word-conditional context distributions p and q, we have the so-called δ-divergences (Zhu and Rohwer, 1998):

$$D_\delta(p,q) := \frac{1 - \sum p^\delta q^{1-\delta}}{\delta(1-\delta)} \quad (1)$$

Divergences D_0 and D_1 are defined as limits as $\delta \to 0$ and $\delta \to 1$:

$$D_1(p,q) = D_0(q,p) = \sum p \log \frac{p}{q}$$

In other words, $D_1(p,q)$ is the KL-divergence of p from q. Members of this divergence family are in some sense preferred by theory to alternative measures. It can be shown that the δ-divergences (or divergences defined by combinations of them, such as the Jensen-Shannon or "skew" divergences (Lee, 1999)) are the only ones that are robust to redundant contexts (i.e., only divergences in this family are *invariant*) (Csiszár, 1975).

Several notions of lexical similarity have been based on the KL-divergence. Note that if any $q_i = 0$, then $D_1(p,q)$ is infinite; in general, the KL-divergence is very sensitive to small probabilities, and careful attention must be paid to smoothing if it is to be used with text co-occurrence data. The Jensen-Shannon divergence—an average of the divergences of p and q from their mean distribution—does not share this sensitivity and has previously been used in tests of lexical similarity (Lee, 1999). Furthermore, unlike the KL-divergence, it is symmetric, presumably a desirable property in this setting, since synonymy is a symmetric relation, and our test design exploits this symmetry.

However, $D_{1/2}(p,q)$, the Hellinger distance[3], is also symmetric and robust to small or zero estimates. To our knowledge, the Hellinger distance has not previously been assessed as a measure of lexical similarity. We experimented with both the Hellinger distance and Jensen-Shannon (JS) divergence, and obtained close scores across a wide range of parameter settings, with the Hellinger yielding a slightly better top score. We report results only for the Hellinger distance below. As will be seen, neither the Hellinger nor the JS divergence are optimal for this task.

In pursuit of synonymy, Ehlert (2003) derives a formula for the probability of the target word given a response word:

$$P(w_1|w_2) = \frac{\sum_i P(w_1|c_i)P(w_2|c_i)P(c_i)}{P(w_2)} \quad (2)$$

$$= P(w_1) \sum_i \frac{P(c_i|w_1)P(c_i|w_2)}{P(c_i)} \quad (3)$$

The second line, which fits more conveniently into our framework, follows from the first (Ehlert's expression) through an application of Bayes Theorem. While this measure falls outside the class of δ-divergences, our experiments confirm its relative strength on synonymy tests.

It is possible to unify the δ-divergences with Ehlert's expression by defining a broader class of measures:

$$D_{\delta,\gamma,\alpha}(p,q) = 1 - \sum_i c_i^{-\alpha} p_i^\delta q_i^\gamma \quad (4)$$

where c_i is the marginal probability of a single context, and p_i and q_i are its respective word-conditional probabilities. Since, in the context of a given question, $P(w_1)$ does not change, maximizing the expression in Equation 3 is the same as minimizing $D_{1,1,1}$. $D_{\delta,(1-\delta),0}$ recovers the δ divergences up to a constant multiple, and $D_{1,1,0}$ provides the complement of the familiar inner-product measure.

[3] Actually, $D_{1/2}(p,q)$ is four times the square of the Hellinger distance.

4 Evaluation

We experimented with various distance measures and context policies using the full North American News corpus. We count approximately one billion words in this corpus, which is roughly four times the size of the largest corpus considered by Ehlert.

Except where noted, the numbers reported here are the result of taking the full WBST, a total of 23,570 test questions. Given this number of questions, scores where most of the results fall are accurate to within plus or minus 0.6% at the 95% confidence level.

4.1 Performance Bounds

In order to provide a point of comparison, the paper's authors each answered the same random sample of 100 questions from each part of speech. Average performance over this sample was 88.4%. The one non-native speaker scored 80.3%. As will be seen, this is better than the best automated result.

The expected score, in the absence of any semantic information, is 25%. However, as noted, target and answer words are more polysemous than decoy words on average, and this can be exploited to establish a higher baseline. Since the frequency of a word is correlated with its polysemy, a strategy which always selects the most frequent word among the response words yields 39.2%, 34.5%, 29.1%, and 38.0% on nouns, verbs, adjectives, and adverbs, respectively, for an average score of 35.2%.

4.2 An Initial Comparison

Table 2 displays a basic comparison of the distance measures and context definitions enumerated so far. For each distance measure (Manhattan, Euclidean, Cosine, Hellinger, and Ehlert), results are shown for window sizes 1 to 4 (columns). Results are further sub-divided according to whether strict direction and distance are false (*None*), only strict direction is true (*Dir*), or both strict direction and strict distance are true (*Dir+Dist*). In bold is the best score, along with any scores indistinguishable from it at the 95% confidence level.

Notable in Table 2 are the somewhat depressed scores, compared with those reported for the TOEFL. Ehlert reports a best score on the TOEFL of 82%, whereas the best we are able to achieve on the WBST is 67.6%. Although there are differences in some of the experimental details (Ehlert employs a triangular window weighting and experiments with stemming), these probably do not account for the discrepancy. Rather, this appears to be a harder test than the TOEFL—despite the fact that all words involved are seen with high frequency.

		Window Size			
		1	2	3	4
Manh	None	54.2	58.8	60.4	60.6
	Dir	54.3	58.5	60.3	60.8
	Dir+Dist	–	57.3	58.8	58.9
Euc	None	42.9	45.3	46.6	47.6
	Dir	43.2	45.7	46.8	47.6
	Dir+Dist	–	44.9	45.3	45.6
Cos	None	44.9	46.7	47.6	48.3
	Dir	46.2	48.0	48.6	49.2
	Dir+Dist	–	48.0	48.4	48.5
Hell	None	57.9	62.3	62.2	61.0
	Dir	57.2	62.6	63.3	61.8
	Dir+Dist	–	61.2	61.7	61.1
Ehl	None	64.0	66.2	66.2	65.7
	Dir	63.9	66.9	**67.6**	**67.1**
	Dir+Dist	–	66.4	**67.2**	**67.5**

Table 2: Accuracy on the WBST: an initial comparison of distance measures and context definitions.

It is hard to escape the conclusion that, in pursuit of high scores, choice of distance measure is more critical than the specific definition of context. All scores returned by the Ehlert metric are significantly higher than any returned by other distance measures. Among the Ehlert scores, there is surprising lack of sensitivity to context policy, given a window of size 2 or larger.

Although the Hellinger distance yields scores only in the middle of the pack, it might be that other divergences from the δ-divergence family, such as the KL-divergence, would yield better scores. We experimented with various settings of δ in Equation 1. In all cases, we observed bell-shaped curves with peaks approximately at $\delta = 0.5$ and locally worst performance with values at or near 0 or 1. This held true when we used maximum likelihood estimates, or under a simple smoothing regime in which

all cells of the co-occurrence matrix were initialized with various fixed values. It is possible that numerical issues are nevertheless partly responsible for the poor showing of the KL-divergence. However, given the symmetry of the synonymy relation, it would be surprising if some value of δ far from 0.5 was ultimately shown to be best.

4.3 The Importance of Weighting

The Ehlert measure and the cosine are closely related—both involve an inner product between vectors—yet they return very different scores in Table 2. There are two differences between these methods, normalization and vector element weighting. We presume that normalization does not account for the large score difference, and attribute the discrepancy, and the general strength of the Ehlert measure, to importance weighting.

In information retrieval, it is common to take the cosine between vectors where vector elements are not raw frequency counts, but counts weighted using some version of the "inverse document frequency" (IDF). We ran the cosine experiment again, this time weighting the count of context i by $log(D/d_i)$, where D is the number of rows in the count matrix N and d_i is the number of rows containing a non-zero count for context i. The results confirmed our expectation. The performance of "CosineIDF" for a window size of 3 with strict direction was 64.0%, which is better than Hellinger but worse than the Ehlert measure. This was the best result returned for "CosineIDF."

4.4 Optimizing Distance Measures

Both the Hellinger distance and the Ehlert measure are members of the family of measures defined by Equation 4. Although there are theoretical reasons to prefer each to neighboring members of the same family (see the discussion following Equation 1), we undertook to validate this preference empirically. We conducted parameter sweeps of α, δ, and γ, first exploring members of the family $\delta = \gamma$, of which both Hellinger and Ehlert are members. Specifically, we explored the space between $\delta = \gamma = 0.5$ and $\delta = \gamma = 1$, first in increments of 0.1, then in increments of 0.01 around the approximate maximum, in all cases varying α widely.

This experiment clearly favored a region midway between the Hellinger and Ehlert measures. We identified $\delta = \gamma = 0.75$, with $\alpha = 1.1$ as the approximate midpoint of this optimal region. We next varied δ and γ independently around this point. This resulted in no improvement to the score, confirming our expectation that some point along $\delta = \gamma$ would be best. For the sake of brevity, we will refer to this best point ($D_{0.75,0.75,1.1}$) as the "Optimal" measure. As Table 3 indicates, this measure is significantly better than the Ehlert measure, or any other measure investigated here.

	Noun	Verb	Adj	Adv	All
Ehlert	71.6	57.2	73.4	72.5	67.6
Optimal	75.8	63.8	76.4	76.6	72.2

Table 3: Comparison between the Ehlert measure and the "optimal" point in the space of measures defined by Equation 4 ($\delta = \gamma = 0.75$, $\alpha = 1.1$), by part of speech. Context policy is window size 3 with strict direction.

This clear separation between Ehlert and Optimal does not hold for the original TOEFL. Using the same context policy, we applied these measures to 298 of the 300 questions used by Ehlert (all questions except those involving multi-word terms, which our framework does not currently support). Optimal returns 84.2%, while Ehlert's measure returns 83.6%, which is slightly better than the 82% reported by Ehlert. The two results are not distinguishable with any statistical significance.

Interesting in Table 3 is the range of scores seen across parts of speech. The variation is even wider under other measures, the usual ordering among parts of speech being (from highest to lowest) adverb, adjective, noun, verb. In Section 4.6, we attempt to shed some light on both this ordering and the close outcome we observe on the TOEFL.

4.5 Optimizing Context Policy

It is certain that not every contextual token seen within the co-occurrence window is equally important to the detection of synonymy, and probable that some such tokens are useless or even detrimental. On the one hand, the many low-frequency events in the tails of the context distributions consume a lot of space, perhaps without contributing much infor-

mation. On the other, very-high-frequency terms are typically closed-class and stop words, possibly too common to be useful in making semantic distinctions. We investigated excluding words at both ends of the frequency spectrum.

We experimented with two kinds of exclusion policies: one excluding the k most frequent terms, for k ranging between 10 and 200; and one excluding terms occurring fewer than k times, for k ranging from 3 up to 100. Both Ehlert and Optimal were largely invariant across all settings; no statistically significant improvements or degradations were observed. Optimal returned scores ranging from 72.0%, when contexts with marginal frequency fewer than 100 were ignored, up to 72.6%, when the 200 most frequent terms were excluded.

Note there is a large qualitative difference between the two exclusion procedures. Whereas we exclude only at most 200 words in the high-frequency experiment, the number of terms excluded in the low-frequency experiment ranges from 939,496 (less than minimum frequency 3) to 1,534,427 (minimum frequency 100), out of a vocabulary containing about 1.6 million terms. Thus, it is possible to reduce the expense of corpus analysis substantially without sacrificing semantic fidelity.

4.6 Polysemy

We hypothesized that the variation in scores across part of speech has to do with the average number of senses seen in a test set. Common verbs, for example, tend to be much more polysemous (and syntactically ambiguous) than common adverbs. WordNet allows us to test this hypothesis.

We define the *polysemy level* of a question as the sum of the number of senses in WordNet of its target and answer words. Polysemy levels in our question set range from 2 up to 116. Calculating the average polysemy level for questions in the various parts of speech—5.1, 6.7, 7.5, and 10.4, for adverbs, adjectives, nouns, and verbs, respectively—provides support for our hypothesis, inasmuch as this ordering aligns with test scores. By contrast, the average polysemy level in the TOEFL, which spans all four parts of speech, is 4.6.

Plotting performance against polysemy level helps explain why Ehlert and Optimal return roughly equivalent performance on the original TOEFL. Figure 1 plots the Ehlert and Optimal measures as a function of the polysemy level of the questions. To produce this plot, we grouped questions according to polysemy level, creating many smaller tests, and scored each measure on each test separately.

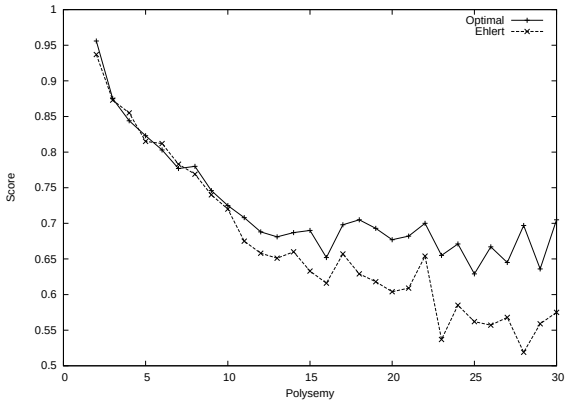


Figure 1: Score as a function of polysemy level.

At low polysemy levels, the Ehlert and Optimal measures perform equally well. The advantage of Optimal over Ehlert appears to lie specifically in its relative strength in handling polysemous terms.

5 Discussion

Specific conclusions regarding the "Optimal" measure are problematic. We do not know whether or to what extent this particular parameter setting is universally best, best only for English, best for newswire English, or best only for the specific test we have devised. We have restricted our attention to a relatively small space of similarity measures, excluding many previously proposed measures of lexical affinity (but see Weeds, et al (2004), and Lee (1999) for some empirical comparisons). Lee observed that measures from the space of invariant divergences (particularly the JS and skew divergences) perform at least as well as any of a wide variety of alternatives. As noted, we experimented with the JS divergence and observed accuracies that tracked those of the Hellinger closely. This provides a point of comparison with the measures investigated by Lee, and recommends both Ehlert's measure and what we have called "Optimal" as credible, perhaps superior alternatives. More generally, our results argue for some form of feature importance

weighting.

Empirically, the strength of Optimal on the WBST is a feature of its robustness in the presence of polysemy. Both Ehlert and Optimal are expressed as a sum of ratios, in which the numerator is a product of some function of conditional context probabilities, and the denominator is some function of the marginal probability. The Optimal exponents on both the numerator and denominator have the effect of advantaging lower-probability events, relative to Ehlert. In our test, WordNet senses are sampled uniformly at random. Perhaps its emphasis on lower probability events allows Optimal to sacrifice some fidelity on high-frequency senses in exchange for increased sensitivity to low-frequency ones.

It is clear, however, that polysemy is a critical hurdle confronting distributional approaches to lexical semantics. Figure 1 shows that, in the absence of polysemy, distributional comparisons detect synonymy quite well. Much of the human advantage over machines on this task may be attributed to an awareness of polysemy. In order to achieve performance comparable to that of humans, therefore, it is probably not enough to optimize context policies or to rely on larger collections of text. Instead, we require strategies for detecting and resolving latent word senses.

Pantel and Lin (2002) propose one such method, evaluated by finding the degree of overlap between sense clusters and synsets in WordNet. The above considerations suggest that a possibly more pertinent test of such approaches is to evaluate their utility in the detection of semantic similarity between specific polysemous terms. We expect to undertake such an evaluation in future work.

Acknowledgments. This material is based on work funded in whole or in part by the U.S. Government. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the U.S. Government.

References

I. Csiszár. 1975. I-divergence geometry of probability distributions and minimization problems. *Annals of Probability*, 3:146–158.

B. Ehlert. 2003. Making accurate lexical semantic similarity judgments using word-context co-occurrence statistics. Master's thesis, University of California, San Diego.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.

Z. Harris. 1968. *Mathematical Structures of Language*. Interscience Publishers, New York.

T.K. Landauer and S.T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

L. Lee. 1999. Measures of distributional similarity. In *Proceedings of the 37th ACL*.

D. Lin. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of ACL-97*, Madrid, Spain.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL98*, Montreal, Canada.

P. Pantel and D. Lin. 2002. Discovering word senses from text. In *Proceedings of KDD-02*, Edmonton, Canada.

M. Sahlgren. 2001. Vector-based semantic analysis: representing word meanings based on random labels. In *Semantic Knowledge Acquisition and Categorisation Workshop, ESSLLI 2001*, Helsinki, Finland.

E. Terra and C.L.A. Clarke. 2003. Frequency estimates for statistical word similarity measures. In *Proceedings of HLT/NAACL 2003*, Edmonton, Canada.

P.D. Turney, M.L. Littman, J. Bigham, and V. Schnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*.

P.D. Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (ECML-01)*.

J. Weeds, D. Weir, and D. McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of CoLing 2004*, Geneva, Switzerland.

H. Zhu and R. Rohwer. 1998. Information geometry, Bayesian inference, ideal estimates, and error decomposition. Technical Report 98-06-045, Santa Fe Institute.

Word Independent Context Pair Classification Model for Word Sense Disambiguation

Cheng Niu, Wei Li, Rohini K. Srihari, and Huifeng Li
Cymfony Inc.
600 Essjay Road, Williamsville, NY 14221, USA.
{cniu, wei, rohini,hli}@cymfony.com

Abstract

Traditionally, word sense disambiguation (WSD) involves a different context classification model for each individual word. This paper presents a weakly supervised learning approach to WSD based on learning a word independent context pair classification model. Statistical models are not trained for classifying the word contexts, but for classifying a pair of contexts, i.e. determining if a pair of contexts of the same ambiguous word refers to the same or different senses. Using this approach, annotated corpus of a target word *A* can be explored to disambiguate senses of a different word *B*. Hence, only a limited amount of existing annotated corpus is required in order to disambiguate the entire vocabulary. In this research, maximum entropy modeling is used to train the word independent context pair classification model. Then based on the context pair classification results, clustering is performed on word mentions extracted from a large raw corpus. The resulting context clusters are mapped onto the external thesaurus WordNet. This approach shows great flexibility to efficiently integrate heterogeneous knowledge sources, e.g. trigger words and parsing structures. Based on Senseval-3 Lexical Sample standards, this approach achieves state-of-the-art performance in the unsupervised learning category, and performs comparably with the supervised Naïve Bayes system.

1 Introduction

Word Sense Disambiguation (WSD) is one of the central problems in Natural Language Processing. The difficulty of this task lies in the fact that context features and the corresponding statistical distribution are different for each individual word. Traditionally, WSD involves training the context classification models for each ambiguous word. (Gale et al. 1992) uses the Naïve Bayes method for context classification which requires a manually annotated corpus for each ambiguous word. This causes a serious *Knowledge Bottleneck*. The bottleneck is particularly serious when considering the domain dependency of word senses. To overcome the *Knowledge Bottleneck*, unsupervised or weakly supervised learning approaches have been proposed. These include the bootstrapping approach (Yarowsky 1995) and the context clustering approach (Schütze 1998).

The above unsupervised or weakly supervised learning approaches are less subject to the *Knowledge Bottleneck*. For example, (Yarowsky 1995) only requires sense number and a few seeds for each sense of an ambiguous word (hereafter called *keyword*). (Schütze 1998) may only need minimal annotation to map the resulting context clusters onto external thesaurus for benchmarking and application-related purposes. Both methods are based on trigger words only.

This paper presents a novel approach based on learning word-independent context pair classification model. This idea may be traced back to (Schütze 1998) where context clusters based on generic Euclidean distance are regarded as distinct word senses. Different from (Schütze 1998), we observe that generic context clusters may not always correspond to distinct word senses. Therefore, we used supervised machine learning to model the relationships between the context distinctness and the sense distinctness.

Although supervised machine learning is used for the context pair classification model, our overall system belongs to the weakly supervised category because the learned context pair classification

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 33–39, Ann Arbor, June 2005.

model is independent of the keyword for disambiguation. Our system does not need human-annotated instances for each target ambiguous word. The weak supervision is performed by using a limited amount of existing annotated corpus which does not need to include the target word set.

The insight is that the correlation regularity between the sense distinction and the context distinction can be captured at Part-of-Speech category level, independent of individual words or word senses. Since context determines the sense of a word, a reasonable hypothesis is that there is some mechanism in the human comprehension process that will decide when two contexts are similar (or dissimilar) enough to trigger our interpretation of a word in the contexts as one meaning (or as two different meanings). We can model this mechanism by capturing the sense distinction regularity at category level.

In the light of this, a maximum entropy model is trained to determine if a pair of contexts of the same keyword refers to the same or different word senses. The maximum entropy modeling is based on heterogeneous context features that involve both trigger words and parsing structures. To ensure the resulting model's independency of individual words, the keywords used in training are different from the keywords used in benchmarking. For any target keyword, a collection of contexts is retrieved from a large raw document pool. Context clustering is performed to derive the optimal context clusters which globally fit the local context pair classification results. Here statistical annealing is used for its optimal performance. In benchmarking, a mapping procedure is required to correlate the context clusters with external ontology senses.

In what follows, Section 2 formulates the maximum entropy model for context pair classification. The context clustering algorithm, including the object function of the clustering and the statistical annealing-based optimization, is described in Section 3. Section 4 presents and discusses benchmarks, followed by conclusion in Section 5.

2 Maximum Entropy Modeling for Context Pair Classification

Given n mentions of a keyword, we first introduce the following symbols. C_i refers to the i-th context. S_i refers to the sense of the i-th context. $CS_{i,j}$ refers to the context similarity between the i-th context and the j-th context, which is a subset of the predefined context similarity features. f_α refers to the α-th predefined context similarity feature. So $CS_{i,j}$ takes the form of $\{f_\alpha\}$.

In this section, we study the context pair classification task, i.e. given a pair of contexts C_i and C_j of the same target word, are they referring to the same sense? This task is formulated as comparing the following conditional probabilities: $\Pr(S_i = S_j | CS_{i,j})$ and $\Pr(S_i \neq S_j | CS_{i,j})$. Unlike traditional context classification for WSD where statistical model is trained for each individual word, our context pair classification model is trained for each Part-of-speech (POS) category. The reason for choosing POS as the appropriate category for learning the context similarity is that the parsing structures, hence the context representation, are different for different POS categories.

The training corpora are constructed using the Senseval-2 English Lexical Sample training corpus. To ensure the resulting model's independency of individual words, the target words used for benchmarking (which will be the ambiguous words used in Senseval-3 English Lexicon Sample task) are carefully removed in the corpus construction process. For each POS category, positive and negative instances are constructed as follows.

Positive instances are constructed using context pairs referring to the same sense of a word. Negative instances are constructed using context pairs that refer to different senses of a word.

For each POS category, we have constructed about 36,000 instances, half positive and half negative. The instances are represented as pairwise context similarities, taking the form of $\{f_\alpha\}$.

Before presenting the context similarity features we used, we first introduce the two categories of the involved context features:

i) Co-occurring trigger words within a predefined window size equal to 50 words to both sides of the keyword. The trigger words are learned from a TIPSTER document pool containing ~170 million words of AP and WSJ news articles. Following (Schütze 1998), χ^2 is used to measure the cohesion between the keyword and a co-occurring word. In our ex-

periment, all the words are first sorted based on its χ^2 with the keyword, and then the top 2,000 words are selected as trigger words.

ii) Parsing relationships associated with the keyword automatically decoded by a broad-coverage parser, with F-measure (i.e. the precision-recall combined score) at about 85% (reference temporarily omitted for the sake of blind review). The logical dependency relationships being utilized are listed below.

Noun: *subject-of,*
object-of,
complement-of,
has-adjective-modifier,
has-noun-modifier,
modifier-of,
possess,
possessed-by,
appositive-of

Verb: *has-subject,*
has-object,
has-complement,
has-adverb-modifier,
has-prepositional-phrase-modifier

Adjective: *modifier-of,*
has-adverb-modifier

Based on the above context features, the following three categories of context similarity features are defined:

(1) VSM-based (Vector Space Model based) trigger word similarity: the trigger words around the keyword are represented as a vector, and the word *i* in context *j* is weighted as follows:

$$weight(i, j) = tf(i, j) * \log \frac{D}{df(i)}$$

where $tf(i, j)$ is the frequency of word *i* in the *j-th* context; D is the number of documents in the pool; and $df(i)$ is the number of documents containing the word *i*. D and $df(i)$ are estimated using the document pool introduced above. The cosine of the angle between two resulting vectors is used as the context similarity measure.

(2) LSA-based (Latent Semantic Analysis based) trigger word similarity: LSA (Deerwester et al. 1990) is a technique used to uncover the underlying semantics based on co-occurrence data. The first step of LSA is to construct word-vs.-document co-occurrence matrix. Then singular value decomposition (SVD) is performed on this co-occurring matrix. The key idea of LSA is to reduce noise or insignificant association patterns by filtering the insignificant components uncovered by SVD. This is done by keeping only the top *k* singular values. By using the resulting word-vs.-document co-occurrence matrix after the filtering, each word can be represented as a vector in the semantic space.

In our experiment, we constructed the original word-vs.-document co-occurring matrix as follows: 100,000 documents from the TIPSTER corpus were used to construct the co-occurring matrix. We processed these documents using our POS tagger, and selected the top *n* most frequently mentioned words from each POS category as base words:

top 20,000 common nouns
top 40,000 proper names
top 10,000 verbs
top 10,000 adjectives
top 2,000 adverbs

In performing SVD, we set *k* (i.e. the number of nonzero singular values) as 200, following the practice reported in (Deerwester et al. 1990) and (Landauer & Dumais, 1997).

Using the LSA scheme described above, each word is represented as a vector in the semantic space. The co-occurring trigger words are represented as a vector summation. Then the cosine of the angle between the two resulting vector summations is computed, and used as the context similarity measure.

(3) LSA-based parsing relationship similarity: each relationship is in the form of $R_\alpha(w)$. Using LSA, each word *w* is represented as a

semantic vector $V(w)$. The similarity between $R_\alpha(w_1)$ and $R_\alpha(w_2)$ is represented as the cosine of the angle between $V(w_1)$ and $V(w_2)$. Two special values are assigned to two exceptional cases: (i) when no relationship R_α is decoded in both contexts; (ii) when the relationship R_α is decoded only for one context.

In matching parsing relationships in a context pair, if only exact node match counts, very few cases can be covered, hence significantly reducing the effect of the parser in this task. To solve this problem, LSA is used as a type of synonym expansion in matching. For example, using LSA, the following word similarity values are generated:

similarity(good, good) 1.00
similarity(good, pretty) 0.79
similarity(good, great) 0.72
……

Given a context pair of a noun keyword, suppose the first context involves a relationship *has-adjective-modifier* whose value is *good,* and the second context involves the same relationship *has-adjective-modifier* with the value *pretty,* then the system assigns 0.79 as the similarity value for this relationship pair.

To facilitate the maximum entropy modeling in the later stage, all the three categories of the resulting similarity values are discretized into 10 integers. Now the pairwise context similarity is represented as a set of similarity features, e.g.

{VSM-Trigger-Words-Similairty-equal-to-2,
LSA-Trigger-Words-Similarity-equal-to-1,
LSA-Subject-Similarity-equal-to-2}.

In addition to the three categories of basic context similarity features defined above, we also define induced context similarity features by combining basic context similarity features using the logical *and* operator. With induced features, the context similarity vector in the previous example is represented as

{VSM-Trigger-Word-Similairty-equal-to-2,
LSA- Trigger-Word-Similarity-equal-to-1,
LSA-Subject-Similarity-equal-to-2,
[VSM-Similairty-equal-to-2 *and*
LSA-Trigger-Word-Similarity-equal-to-1],
[VSM-Similairty-equal-to-2 *and*
LSA-Subject-Similarity-equal-to-2],
……
[VSM-Trigger-Word-Similairty-equal-to-2
and LSA-Trigger-Word-Similarity-equal-to-1
and LSA-Subject-Similarity-equal-to-2]
}

The induced features provide direct and fine-grained information, but suffer from less sampling space. Combining basic features and induced features under a smoothing scheme, maximum entropy modeling may achieve optimal performance.

Using the context similarity features defined above, the training corpora for the context pair classification model is in the following format:

Instance_0 tag="positive" {VSM-Trigger-Word-Similairty-equal-to-2, …}
Instance_1 tag="negative" {VSM-Trigger-Word-Similairty-equal-to-0, …}
……
where *positive* tag denotes a context pair associated with same sense, and *negative* tag denotes a context pair associated with different senses.

The maximum entropy modeling is used to compute the conditional probabilities $\Pr(S_i = S_j | CS_{i,j})$ and $\Pr(S_i \neq S_j | CS_{i,j})$: once the context pair $CS_{i,j}$ is represented as $\{f_\alpha\}$, the conditional probability is given as

$$\Pr(t|\{f_\alpha\}) = \frac{1}{Z}\prod_{f\in\{f_\alpha\}} w_{t,f} \qquad (1)$$

where $t \in \{S_i = S_j, S_i \neq S_j\}$, Z is the normalization factor, $w_{t,f}$ is the weight associated with tag t and feature f. Using the training corpora constructed above, the weights can be computed based on Iterative Scaling algorithm (Pietra etc. 1995) The exponential prior smoothing scheme (Goodman 2003) is adopted in the training.

3 Context Clustering based on Context Pair Classification Results

Given n mentions $\{C_i\}$ of a keyword, we use the following context clustering scheme. The discovered context clusters correspond to distinct word senses.

For any given context pair, the context similarity features defined in Section 2 are computed. With n mentions of the same keyword, $\frac{n(n-1)}{2}$ context similarities $CS_{i,j}\ (i \in [1,n], j \in [1,i))$ are computed. Using the context pair classification model, each pair is associated with two scores $sc^0_{i,j} = \log(\Pr(S_i = S_j | CS_{i,j}))$ and $sc^1_{i,j} = \log(\Pr(S_i = S_j | CS_{i,j}))$ which correspond to the probabilities of two situations: the pair refers to the same or different word senses.

Now we introduce the symbol $\{K, M\}$ which refers to the final context cluster configuration, where K refers to the number of distinct sense, and M represents the many-to-one mapping (from contexts to a sense) such that $\mathrm{M(i)} = \mathrm{j}, \mathrm{i} \in [1,\mathrm{n}], \mathrm{j} \in [1,\mathrm{K}]$. Based on the pairwise scores $\{sc^0_{i,j}\}$ and $\{sc^1_{i,j}\}$, WSD is formulated as searching for $\{K, M\}$ which maximizes the following global scores:

$$sc(\{\mathrm{K},\mathrm{M}\}) = \sum_{\substack{i\in[1,n],\\ j\in[1,i)}} sc_{i,j}^{k(i,j)} \quad (2)$$

where $k(i,j) = \begin{cases} 0, if\ M(i) = M(j) \\ 1, \quad otherwise \end{cases}$

Similar clustering scheme has been used successfully for the task of co-reference in (Luo etc. 2004), (Zelenko, Aone and Tibbetts, 2004a) and (Zelenko, Aone and Tibbetts, 2004b).

In this paper, statistical annealing-based optimization (Neal 1993) is used to search for $\{K, M\}$ which maximizes Expression (2).

The optimization process consists of two steps. First, an intermediate solution $\{K, M\}_0$ is computed by a greedy algorithm. Then by setting $\{K, M\}_0$ as the initial state, statistical annealing is applied to search for the global optimal solution. The optimization algorithm is as follows.

1. Set the initial state $\{K, M\}$ as $K = n$, and $M(i) = i,\ \mathrm{i} \in [1,\mathrm{n}]$;
2. Select a cluster pair for merging that maximally increases $sc(\{\mathrm{K},\mathrm{M}\}) = \sum_{\substack{i\in[1,n],\\ j\in[1,i)}} sc_{i,j}^{k(i,j)}$
3. If no cluster pair can be merged to increase $sc(\{\mathrm{K},\mathrm{M}\}) = \sum_{\substack{i\in[1,n],\\ j\in[1,i)}} sc_{i,j}^{k(i,j)}$, output $\{K, M\}$ as the intermediate solution; otherwise, update $\{K, M\}$ by the merge and go to step 2.

Using the intermediate solution $\{K, M\}_0$ of the greedy algorithm as the initial state, the statistical annealing is implemented using the following pseudo-code:

```
Set {K,M} = {K,M}_0 ;
for(β = β_0 ; β < β_final ; β* = 1.01)
{
  iterate pre-defined number of times
  {
    set {K,M}_1 = {K,M};
    update {K,M}_1 by randomly changing
  cluster number and cluster contents;
      set x = sc({K,M}_1) / sc({K,M})
      if(x>=1)
      {
        set {K,M} = {K,M}_1
      }
      else
      {
        set {K,M} = {K,M}_1 with probability
          x^β .
      }
      if sc({K,M}) > sc({K,M}_0)
      then set {K,M}_0 = {K,M}
  }
}
output {K,M}_0 as the optimal state.
```

4 Benchmarking

Corpus-driven context clusters need to map to a word sense standard to facilitate performance benchmark. Using Senseval-3 evaluation standards, we implemented the following procedure to map the context clusters:

i) Process TIPSTER corpus and the original unlabeled Senseval-3 corpora (including the training corpus and the testing corpus) by our parser, and save all the parsing results into a repository.

ii) For each keyword, all related contexts in Senseval-3 corpora and up-to-1,000 related contexts in TIPSTER corpus are retrieved from the repository.

iii) All the retrieved contexts are clustered based on the context clustering algorithm presented in Sect. 2 and 3.

iv) For each keyword sense, three annotated contexts from Senseval-3 training corpus are used for the sense mapping. The context cluster is mapped onto the most frequent word sense associated with the cluster members. By design, the context clusters correspond to distinct senses, therefore, we do not allow multiple context clusters to be mapped onto one sense. In case multiple clusters correspond to one sense, only the largest cluster is retained.

v) Each context in the testing corpus is tagged with the sense to which its context cluster corresponds to.

As mentioned above, Sensval-2 English lexical sample training corpora is used to train the context pair classification model. And Sensval-3 English lexical sample testing corpora is used here for benchmarking. There are several keyword occurring in both Senseval-2 and Senseval-3 corpora. The sense tags associated with these keywords are not used in the context pair classification training process.

In order to gauge the performance of this new weakly supervised learning algorithm, we have also implemented a supervised Naïve Bayes system following (Gale et al. 1992). This system is trained based on the Senseval-3 English Lexical Sample training corpus. In addition, for the purpose of quantifying the contribution from the parsing structures in WSD, we have run our new system with two configurations: (i) using only trigger words; (ii) using both trigger words and parsing relationships. All the benchmarking is performed using the Senseval-3 English Lexical Sample testing corpus and standards.

The performance benchmarks for the two systems in three runs are shown in Table 1, Table 2 and Table 3. When using only trigger words, this algorithm has 8 percentage degradation from the supervised Naïve Bayes system (see Table 1 vs. Table 2). When adding parsing structures, performance degradation is reduced, with about 5 percentage drop (see Table 3 vs. Table 2). Comparing Table 1 with Table 3, we observe about 3% enhancement due to the contribution from the parsing support in WSD. The benchmark of our algorithm using both trigger words and parsing relationships is one of the best in unsupervised category of the Senseval-3 Lexical Sample evaluation.

Table 1. New Algorithm Using Only Trigger Words

Category	Accuracy	
	Fine grain (%)	Coarse grain (%)
Adjective (5)	46.3	60.8
Noun (20)	54.6	62.8
Verb (32)	54.1	64.2
Overall	54.0	63.4

Table 2. Supervised Naïve Bayes System

Category	Accuracy	
	Fine grain (%)	Coarse grain (%)
Adjective (5)	44.7	56.6
Noun (20)	66.3	74.5
Verb (32)	58.6	70.0
Overall	61.6	71.5

Table 3. New Algorithm Using Both Trigger Words and Parsing

Category	Accuracy	
	Fine grain (%)	Coarse grain (%)
Adjective (5)	49.1	64.8
Noun (20)	57.9	66.6
Verb (32)	55.3	66.3
Overall	56.3	66.4

It is noted that Naïve Bayes algorithm has many variation, and its performance has been greatly enhanced during recent research. Based on Senseval-3 results, the best Naïve Bayse system outperform our version (which is implemented based on Gale et al. 1992) by 8%~10%. So the best supervised WSD systems output-perform our weakly supervised WSD system by 13%~15% in accuracy.

5 Conclusion

We have presented a weakly supervised learning approach to WSD. Statistical models are not trained for the contexts of each individual word, but for context pair classification. This approach overcomes the knowledge bottleneck that challenges supervised WSD systems which need labeled data for each individual word. It captures the correlation regularity between the sense distinction and the context distinction at Part-of-Speech category level, independent of individual words and senses. Hence, it only requires a limited amount of existing annotated corpus in order to disambiguate the full target set of ambiguous words, in particular, the target words that do not appear in the training corpus.

The weakly supervised learning scheme can combine trigger words and parsing structures in supporting WSD. Using Senseval-3 English Lexical Sample benchmarking, this new approach reaches one of the best scores in the unsupervised category of English Lexical Sample evaluation. This performance is close to the performance for the supervised Naïve Bayes system.

In the future, we will implement a new scheme to map context clusters onto WordNet senses by exploring WordNet glosses and sample sentences. Based on the new sense mapping scheme, we will benchmark our system performance using Senseval English all-words corpora.

References

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by Latent Semantic Analysis. In *Journal of the American Society of Information Science*

Gale, W., K. Church, and D. Yarowsky. 1992. A Method for Disambiguating Word Senses in a Large Corpus. *Computers and the Humanities*, 26.

Goodman, J. 2003. Exponential Priors for Maximum Entropy Models. In *Proceedings of HLT-NAACL 2004*.

Landauer, T. K., & Dumais, S. T. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104, 211-240, 1997.

Luo, X., A. Ittycheriah, H. Jing, N. Kambhatla and S. Roukos. A Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree. In *The Proceedings of ACL 2004.*

Neal, R.M. 1993. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report, Univ. of Toronto.

Pietra, S. D., V. D. Pietra, and J. Lafferty. 1995. Inducing Features Of Random Fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Schütze, H. 1998. Automatic Word Sense Disambiguation. *Computational Linguistics,* 23.

Yarowsky, D. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of ACL 1995.*

Zelenko, D., C. Aone and J. 2004. Tibbetts. Coreference Resolution for Information Extraction. In *Proceedings of ACL 2004 Workshop on Reference Resolution and its Application*.

Zelenko, D., C. Aone and J. 2004. Tibbetts. Binary Integer Programming for Information Extraction. In *Proceedings of ACE 2004 Evaluation Workshop*.

Computing Word Similarity and Identifying Cognates with Pair Hidden Markov Models

Wesley Mackay and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{wesleym,kondrak}@cs.ualberta.ca

Abstract

We present a system for computing similarity between pairs of words. Our system is based on Pair Hidden Markov Models, a variation on Hidden Markov Models that has been used successfully for the alignment of biological sequences. The parameters of the model are automatically learned from training data that consists of word pairs known to be similar. Our tests focus on the identification of cognates — words of common origin in related languages. The results show that our system outperforms previously proposed techniques.

1 Introduction

The computation of surface similarity between pairs of words is an important task in many areas of natural language processing. In historical linguistics phonetic similarity is one of the clues for identifying *cognates*, that is, words that share a common origin (Oakes, 2000). In statistical machine translation, cognates are helpful in inducing translation lexicons (Koehn and Knight, 2001; Mann and Yarowsky, 2001), sentence alignment (Melamed, 1999), and word alignment (Tiedemann, 2003). In dialectology, similarity is used for estimating distance between dialects (Nerbonne, 2003). Other applications include cross-lingual information retrieval (Pirkola et al., 2003), detection of confusable drug names (Kondrak and Dorr, 2004), and lexicography (Brew and McKelvie, 1996).

Depending on the context, strong word similarity may indicate either that words share a common origin (*cognates*), a common meaning (*synonyms*), or are related in some way (e.g. *spelling variants*). In this paper, we focus on cognates. Genetic cognates are well-suited for testing measures of word similarity because they arise by evolving from a single word in a proto-language. Unlike rather indefinite concepts like synonymy or confusability, cognation is a binary notion, which in most cases can be reliably determined.

Methods that are normally used for computing word similarity can be divided into orthographic and phonetic. The former includes string edit distance (Wagner and Fischer, 1974), longest common subsequence ratio (Melamed, 1999), and measures based on shared character *n*-grams (Brew and McKelvie, 1996). These usually employ a binary identity function on the level of character comparison. The phonetic approaches, such as Soundex (Hall and Dowling, 1980) and Editex (Zobel and Dart, 1996), attempt to take advantage of the phonetic characteristics of individual characters in order to estimate their similarity. All of the above methods are static, in the sense of having a fixed definition that leaves little room for adaptation to a specific context. In contrast, the methods proposed by Tiedemann (1999) automatically construct weighted string similarity measures on the basis of string segmentation and bitext co-occurrence statistics.

We have created a system for determining word similarity based on a Pair Hidden Markov Model. The parameters of the model are automatically learned from training data that consists of word

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 40–47, Ann Arbor, June 2005.

pairs that are known to be similar. The model is trained using the Baum-Welch algorithm (Baum et al., 1970). We examine several variants of the model, which are characterized by different training techniques, number of parameters, and word length correction method. The models are tested on a cognate recognition task across word lists representing several Indo-European languages. The experiments indicate that our system substantially outperforms the most commonly used approaches.

The paper is organized as follows. Section 2 gives a more detailed description of the problem of word similarity. Section 3 contains an introduction to Pair Hidden Markov Models, while section 4 describes their adaptation to our domain. Sections 5 and 6 report experimental set-up and results.

2 Word Similarity

Word similarity is, at its core, an alignment task. In order to determine similarity between two words, we look at the various alignments that can exist between them. Each component of the alignment is assigned a probability-based score by our trained model. The scores are then combined to produce the overall similarity score for any word pair, which can be used to rank the word pairs against each other. Alternatively, a discrete cut-off point can be selected in order to separate pairs that show the required similarity from the ones that do not.

Before we can align words, they must be separated into symbols. Typically, the symbols are characters in the orthographic representation, and phonemes in the phonetic representation. We also need to put some restrictions on the possible alignments between these symbols. By adopting the following two assumptions, we are able to fully exploit the simplicity and efficiency of the Pair Hidden Markov Model.

First, we assume that the basic ordering of symbols remains the same between languages. This does not mean that every symbol has a corresponding one in the other language, but instead that word transformation comes from three basic operations: *substitution*, *insertion* and *deletion*. Exceptions to this rule certainly exist (e.g. *metathesis*), but are sufficiently infrequent to make the benefits of this constraint far outweigh the costs.

Second, we assume that each symbol is aligned to at most one symbol in the other word. This assumption is aimed at reducing the number of parameters that have to be learned from limited-size training data. If there is a many-to-one correspondence that is consistent between languages, it would be beneficial to change the word representation so that the many symbols are considered as a single symbol instead. For example, a group of characters in the orthographic representation may correspond to a single phoneme if the word is written phonetically.

3 Pair Hidden Markov Models

Hidden Markov Models have been applied successfully to a number of problems in natural language processing, including speech recognition (Jelinek, 1999) and statistical machine translation (Och and Ney, 2000). One of the more intangible aspects of a Hidden Markov Model is the choice of the model itself. While algorithms exist to train the parameters of the model so that the model better describes its data, there is no formulaic way to create the model. We decided to adopt as a starting point a model developed in a different field of study.

Durbin et al. (1998) created a new type of Hidden Markov Model that has been used for the task of aligning biological sequences (Figure 1). Called a Pair Hidden Markov Model, it uses two output streams in parallel, each corresponding to a sequence that is being aligned.[1] The alignment model has three states that represent the basic edit operations: substitution (represented by state "M"), insertion ("Y"), and deletion ("X"). "M", the *match state*, emits an aligned pair of symbols (not necessarily identical) with one symbol on the top and the other on the bottom output stream. "X" and "Y", the *gap states*, output a symbol on only one stream against a gap on the other. Each state has its own emission probabilities representing the likelihood of producing a pairwise alignment of the type described by the state. The model has three transition parameters: δ, ε, and τ. In order to reduce the number of parameters, there is no explicit start state. Rather, the probability of starting in a given state is equal to

[1] Pair Hidden Markov Models have been used in the area of natural language processing once before: Clark (2001) applied PHMMs to the task of learning stochastic finite-state transducers for modeling morphological paradigms.

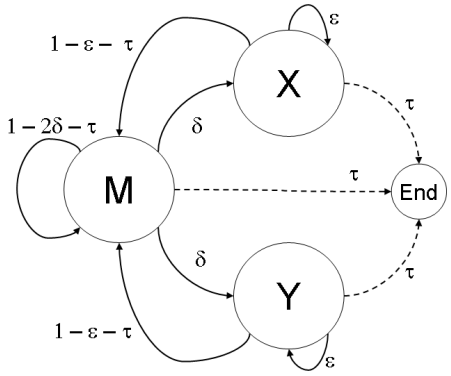


Figure 1: A Pair Hidden Markov Model for aligning biological sequences.

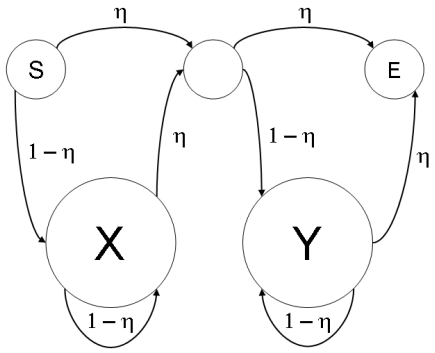


Figure 2: The random Pair Hidden Markov Model.

the probability of going from the match state to the given state.

Durbin et al. (1998) describe several different algorithms that can be used to score and rank paired biological sequences. Two of them are based on common HMM algorithms. The Viterbi algorithm uses the most probable path through the model to score the pair. The forward algorithm computes the total overall probability for a pair by summing up the probabilities of every possible alignment between the words. A third algorithm (the *log odds* algorithm) was designed to take into account how likely the pair would be to occur randomly within the two languages by considering a separately trained *random model* (Figure 2) in conjunction with the similarity model. In the random model, the sequences are assumed to have no relationship to each other, so there is no match state. The log odds algorithm calculates a score for a pair of symbols by dividing the probability of a genuine correspondence between a pair of symbols (the similarity model) by the probability of them co-occurring by chance (the random model). These individual scores are combined to produce an overall score for the pair of sequences in the same way as individual symbol probabilities are combined in other algorithms.

4 PHMMs for Word Similarity

Because of the differences between biological sequence analysis and computing word similarity, the bioinformatics model has to be adapted to handle the latter task. In this section, we propose a number of modifications to the original model and the corresponding algorithms. The modified model is shown in Figure 3.

First, the original model's assumption that an insertion followed by a deletion is the same as a substitution is problematic in the context of word similarity. Covington (1998) illustrates the problem with an example of Italian "due" and the Spanish "dos", both of which mean "two". While there is no doubt that the first two pairs of symbols should be aligned, there is no historical connection between the Italian "e" and the Spanish "s". In this case, a sequence of an insertion and a deletion is more appropriate than a substitution. In order to remedy this problem, we decided to a add a pair of transitions between states "X" and "Y", which is denoted by λ in Figure 3.

The second modification involves splitting the parameter τ into two separate values: τ_M for the match state, and τ_{XY} for the gap states. The original biological model keeps the probability for the transition to the end state constant for all other states. For cognates, and other word similarity tasks, it may be that similar words are more or less likely to end in gaps or matches. The modification preserves the symmetry of the model while allowing it to capture how likely a given operation is to occur at the end of an alignment.

4.1 Algorithm Variations

We have investigated several algorithms for the alignment and scoring of word pairs. Apart from the standard Viterbi (abbreviated **VIT**) and forward (**FOR**) algorithms, we considered two variations of the log odds algorithm, The original log odds algorithm (**LOG**) functions much like a Viterbi algo-

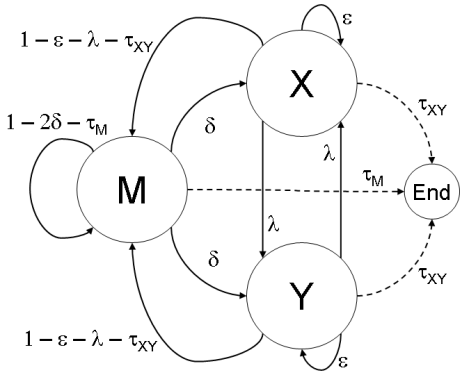


Figure 3: A Pair Hidden Markov Model for aligning words.

rithm, looking at only the most probable sequence of states. We also created another variation, forward log odds (**FLO**), which uses a forward approach instead, considering the aggregate probability of all possible paths through both models.

4.2 Model Variations

Apart from comparing the effectiveness of different algorithms, we are also interested in establishing the optimal structure of the underlying model. The similarity model can be broken up into three sets of parameters: the match probabilities, the gap probabilities, and the transition probabilities. Our goal is to examine the relative contribution of various components of the model, and to find out whether simplifying the model affects the overall performance of the system. Since the match probabilities constitute the core of the model, we focus on the remaining emission and transition probabilities. We also investigate the necessity of including an explicit end state in the model.

The first variation concerns the issue of gap emission probabilities. For the log odds algorithm, Durbin et al. (1998) allow the gap emission probabilities of both the similarity and random models to be equal. While this greatly simplifies the calculations and allows for the emphasis to be on matched symbols, it might be more in spirit with the word similarity task to keep the emissions of the two models separate. If we adopt such an approach, the similarity model learns the gap emission probabilities using the forward-backward algorithm, just as is done with the match probabilities, but the random model uses letter frequencies from the training data instead. A similar test of the effectiveness of trained gap parameters can be performed for the Viterbi and forward algorithms by proceeding in the opposite direction. Instead of deriving the gap probabilities from the training data (as in the original model), we can set them to uniform values after training, thus making the final scores depend primarily on matches.

The second variation removes the effect the transition parameters have on the final calculation. In the resulting model, a transition probability from any state to any state (except the end state) is constant, effectively merging “X”, “Y”, and “M” into a single state. One of the purposes of the separated states was to allow for affine gap penalties, which is why there are different transition parameters for going to a gap state and for staying in that state. By making the transitions constant, we are also taking away the affine gap structure. As a third variant, we try both the first and second variation combined.

The next variation concerns the effect of the end state on the final score. Unlike in the alignment of biological sequences, word alignment boundaries are known beforehand, so an end state is not strictly necessary. It is simple enough to remove the end state from our model after the training has been completed. The remaining transition probability mass is shifted to the transitions that lead to the match state.

Once the end state is removed, it is possible to reduce the number of transition parameters to a single one, by taking advantage of the symmetry between the insertion and deletion states. In the resulting model, the probability of entering a gap state is equal to $\frac{1-x}{2}$, where x is the probability of a transition to the match state. Naturally, the log odds algorithms also have a separate parameter for the random model.

4.3 Correcting for Length

Another problem that needs to be addressed is the bias introduced by the length of the words. The principal objective of the bioinformatics model is the optimal alignment of two sequences. In our case, the alignment is a means to computing word similarity. In fact, some of the algorithms (e.g. the forward algorithm) do not yield an explicit best alignment. While the log odds algorithms have a built-in length correction, the Viterbi and the forward do not.

These algorithms continually multiply probabilities together every time they process a symbol (or a symbol pair), which means that the overall probability of an alignment strongly depends on word lengths. In order to rectify this problem, we multiply the final probability by $\frac{1}{C^n}$, where n is the length of the longer word in the pair, and C is a constant. The value of C can be established on a held-out data set.[2]

4.4 Levenshtein with Learned Weights

Mann and Yarowsky (2001) investigated the induction of translation lexicons via bridge languages. Their approach starts with a dictionary between two well studied languages (e.g. English-Spanish). They then use cognate pairs to induce a *bridge* between two strongly related languages (e.g. Spanish and Italian), and from this create a smaller translation dictionary between the remaining two languages (e.g. English and Italian). They compared the performances of several different cognate similarity (or distance) measures, including one based on the Levenshtein distance, one based on the stochastic transducers of Ristad and Yianilos (1998), and a variation of a Hidden Markov Model. Somewhat surprisingly, the Hidden Markov Model falls well short of the baseline Levenshtein distance.[3]

Mann and Yarowsky (2001) developed yet another model, which outperformed all other similarity measures. In the approach, which they call "Levenshtein with learned weights", the probabilities of their stochastic transducer are transformed into substitution weights for computing Levenshtein distance: 0.5 for highly similar symbols, 0.75 for weakly similar symbols, etc. We have endeavored to emulate this approach (abbreviated **LLW**) by converting the log odds substitution scores calculated from the fully trained model into the substitution weights used by the authors.

[2]Another common method to correct for length is to take the n^{th} root of the final calculation, where n is the length of the longest word. However, our initial experiments indicated that this method does not perform well on the word similarity task.

[3]The HMM model of (Mann and Yarowsky, 2001) is of distinctly different design than our PHMM model. For example, the emission probabilities corresponding to the atomic edit operations sum to one for *each* alphabet symbol. In our model, the emission probabilities for different symbols are interdependent.

5 Experimental Setup

We evaluated our word similarity system on the task of the identification of cognates. The input consists of pairs of words that have the same meaning in distinct languages. For each pair, the system produces a score representing the likelihood that the words are cognate. Ideally, the scores for true cognate pairs should always be higher than scores assigned to unrelated pairs. For binary classification, a specific score threshold could be applied, but we defer the decision on the precision-recall trade-off to downstream applications. Instead, we order the candidate pairs by their scores, and evaluate the ranking using *11-point interpolated average precision* (Manning and Schutze, 2001).

Word similarity is not always a perfect indicator of cognation because it can also result from lexical borrowing and random chance. It is also possible that two words are cognates and yet exhibit little surface similarity. Therefore, the upper bound for average precision is likely to be substantially lower than 100%.

5.1 Data

Training data for our cognate recognition model comes from the Comparative Indoeuropean Data Corpus (Dyen et al., 1992). The data contains word lists of 200 basic meanings representing 95 speech varieties from the Indoeuropean family of languages. Each word is represented in an orthographic form without diacritics using the 26 letters of the Roman alphabet. All cognate pairs are also identified in the data.

The development set[4] consisted of two language pairs: Italian and Serbo-Croatian, as well as Polish and Russian. We chose these two language pairs because they represent very different levels of relatedness: 25.3% and 73.5% of the word pairs are cognates, respectively. The percentage of cognates within the data is important, as it provides a simple baseline from which to compare the success of our algorithms. If our cognate identification process

[4]Several parameters used in our experiments were determined during the development of the word similarity model. These include the random model's parameter η, the constant transition probabilities in the simplified model, and the constant C for correcting the length bias in the Viterbi and forward algorithms. See (Mackay, 2004) for complete details.

were random, we would expect to get roughly these percentages for our recognition precision (on average).

The test set consisted of five 200-word lists representing English, German, French, Latin, and Albanian, compiled by Kessler (2001). The lists for these languages were removed from the training data (except Latin, which was not part of the training set), in order to keep the testing and training data as separate as possible.[5] We converted the test data to have the same orthographic representation as the training data.

5.2 Significance tests

We performed pairwise statistical significance tests for various model and algorithm combinations. Following the method proposed by Evert (2004), we applied Fisher's exact test to counts of word pairs that are accepted by only one of the two tested algorithms. For a given language pair, the cutoff level was set equal to the actual number of cognate pairs in the list. For example, since 118 out of 200 word pairs in the English/German list are cognate, we considered the true and false positives among the set of 118 top scoring pairs. For the overall average of a number of different language pairs, we took the union of the individual sets. For the results in Tables 1 and 2, the pooled set contained 567 out of 2000 pairs, which corresponds to the proportion of cognates in the entire test data (28.35%).

6 Experimental Results

In this section, we first report on the effect of model variations on the overall performance, and then we compare the best results for each algorithm.

6.1 Model Variations

Table 1 shows the average cognate recognition precision on the test set for a number of model variations combined with four basic algorithms, **VIT**, **FOR**, **LOG**, and **FLO**, which were introduced in Section 4.1. The first row refers to the fully trained model without changes. The remaining rows contain the results for the model variations described in Section 4.2. In all cases, the simplifications are in effect during testing only, after the full model had been trained. We also performed experiments with the model simplified prior to training but their results were consistently lower than the results presented here.

Model Variation	Algorithm			
	VIT	FOR	LOG	FLO
full model	0.630	0.621	0.656	**0.631**
gaps const	0.633	0.631	*0.684*	0.624
trans const	0.565	0.507	*0.700*	0.550
both const	0.566	0.531	***0.704***	0.574
no end state	0.626	0.620	0.637	0.601
single param	**0.647**	**0.650**	*0.703*	0.596

Table 1: Average cognate recognition precision for each model and algorithm combination.

With the exception of the forward log odds algorithm, the best results are obtained with simplified models. The model with only a single transition parameter performs particularly well. On the other hand, the removal of the end state invariably causes a decrease in performance with respect to the full model. If a non-essential part of the model is made constant, only the Viterbi-based log odds algorithm improves significantly; the performance of the other three algorithms either deteriorates or shows no significant difference.

Overall, the top four variations of the Viterbi-based log odds algorithm (shown in italics in Table 1) significantly outperform all other PHMM variations and algorithms. This is not entirely unexpected as **LOG** is a more complex algorithms than both **VIT** and **FOR**. It appears that the incorporation of the random model allows **LOG** to better distinguish true similarity from chance similarity. In addition, the log odds algorithms automatically normalize the results based on the lengths of the words under examination. However, from the rather disappointing performance of **FLO**, we conclude that considering all possible alignments does not help the log odds approach.

[5] The complete separation of training and testing data is difficult to achieve in this case because of the similarity of cognates across languages in the same family. For each of the removed languages, there are other closely related languages that are retained in the training set, which may exhibit similar or even identical correspondences.

Languages		Proportion of Cognates	Method						
			LCSR	LLW	ALINE	VIT	FOR	LOG	FLO
English	German	0.590	0.895	0.917	0.916	0.932	0.932	0.930	0.929
French	Latin	0.560	0.902	0.893	0.863	0.916	0.914	0.934	0.904
English	Latin	0.290	0.634	0.713	0.725	0.789	0.792	0.803	0.755
German	Latin	0.290	0.539	0.647	0.706	0.673	0.666	0.730	0.644
English	French	0.275	0.673	0.725	0.615	0.751	0.757	0.812	0.725
French	German	0.245	0.568	0.591	0.504	0.556	0.559	0.734	0.588
Albanian	Latin	0.195	0.541	0.510	0.618	0.546	0.557	0.680	0.541
Albanian	French	0.165	0.486	0.444	0.612	0.505	0.530	0.653	0.545
Albanian	German	0.125	0.275	0.340	0.323	0.380	0.385	0.379	0.280
Albanian	English	0.100	0.245	0.322	0.277	0.416	0.406	0.382	0.403
AVERAGE		0.2835	**0.576**	**0.610**	**0.616**	**0.647**	**0.650**	**0.704**	**0.631**

Table 2: Average cognate recognition precision for various models and algorithms.

6.2 Comparison

Table 2 contains the results of the best variants, which are shown in boldface in Table 1, along with other methods for comparison. The results are separated into individual language pairs from the test set. For the baseline method, we selected the Longest Common Subsequence Ratio (**LCSR**), a measure of orthographic word similarity often used for cognate identification (Brew and McKelvie, 1996; Melamed, 1999; Koehn and Knight, 2001). The LCSR of two words is computed by dividing the length of their longest common subsequence by the length of the longer word. **LLW** stands for "Levenshtein with learned weights", which is described in Section 4.4. We also include the results obtained by the **ALINE** word aligner (Kondrak, 2000) on phonetically-transcribed word lists.

Because of the relatively small size of the lists, the differences among results for individual language pairs are not statistically significant in many cases. However, when the average over all language pairs is considered, the Viterbi-based log odds algorithm (**LOG**) is significantly better than all other algorithms in Table 2. The differences between the remaining algorithms are not statistically significant, except that they all significantly outperform the LCSR baseline.

The fact that **LOG** is significantly better than ALINE demonstrates that given a sufficiently large training set, an HMM-based algorithm can automatically learn the notion of phonetic similarity, which is incorporated into ALINE. ALINE does not involve extensive supervised training, but it requires the words to be in a phonetic, rather than orthographic form. We conjecture that the performance of **LOG** would further improve if it could be trained on phonetically-transcribed multilingual data.

7 Conclusion

We created a system that learns to recognize word pairs that are similar based on some criteria provided during training, and separate such word pairs from those that do not exhibit such similarity or whose similarity exists solely by chance. The system is based on Pair Hidden Markov Models, a technique adapted from the field of bioinformatics. We tested a number of training algorithms and model variations on the task of identifying cognates. However, since it does not rely on domain-specific knowledge, our system can be applied to any task that requires computing word similarity, as long as there are examples of words that would be considered similar in a given context.

In the future, we would like to extend our system by removing the one-to-one constraint that requires alignments to consist of single symbols. It would also be interesting to test the system in other applications, such as the detection of confusable drug names or word alignment in bitexts.

Acknowledgments

This research was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Alberta Informatics Circle of Research Excellence (iCORE).

References

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic function of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

Chris Brew and David McKelvie. 1996. Word-pair extraction for lexicography. In *Proceedings of the 2nd International Conference on New Methods in Language Processing*, pages 45–55.

Alexander Clark. 2001. Learning morphology with Pair Hidden Markov Models. In *Proceedings of the Student Workshop at ACL 2001*.

Michael A. Covington. 1998. Alignment of multiple languages for historical comparison. In *Proceedings of COLING-ACL'98*, pages 275–280.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological sequence analysis*. Cambridge University Press.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5).

Stefan Evert. 2004. Significance tests for the evaluation of ranking methods. In *Proceedings of COLING 2004*, pages 945–951.

Patrick A. V. Hall and Geoff R. Dowling. 1980. Approximate string matching. *Computing Surveys*, 12(4):381–402.

Frederick Jelinek. 1999. *Statistical Methods for Speech Recognition*. The Massachusetts Institute of Technology Press.

Brett Kessler. 2001. *The Significance of Word Lists*. Stanford: CSLI Publications.

Philipp Koehn and Kevin Knight. 2001. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35.

Grzegorz Kondrak and Bonnie Dorr. 2004. Identification of confusable drug names: A new approach and evaluation methodology. In *Proceedings of COLING 2004*, pages 952–958.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000*, pages 288–295.

Wesley Mackay. 2004. Word similarity using Pair Hidden Markov Models. Master's thesis, University of Alberta.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL 2001*, pages 151–158.

Christopher D. Manning and Hinrich Schutze. 2001. *Foundations of Statistical Natural Language Processing*. The MIT Press.

I. Dan Melamed. 1999. Bitext maps and alignment via pattern recognition. *Computational Linguistics*, 25(1):107–130.

John Nerbonne. 2003. Linguistic variation and computation. In *Proceedings of EACL-03*, pages 3–10.

Michael P. Oakes. 2000. Computer estimation of vocabulary in protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics*, 7(3):233–243.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL-2000*, pages 440–447.

Ari Pirkola, Jarmo Toivonen, Heikki Keskustalo, Kari Visala, and Kalervo Jarvelin. 2003. Fuzzy translation of cross-lingual spelling variants. In *Proceedings of SIGIR'03*, pages 345–352.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):522–532.

Jörg Tiedemann. 1999. Automatic construction of weighted string similarity measures. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, Maryland.

Jörg Tiedemann. 2003. Combining clues for word alignment. In *Proceedings of the 10th Conference of the European Chapter of the ACL (EACL03)*.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.

Justin Zobel and Philip Dart. 1996. Phonetic string matching: Lessons from information retrieval. In *Proceedings of SIGIR'96*, pages 166–172.

A Bayesian mixture model for term re-occurrence and burstiness

Avik Sarkar[1], **Paul H Garthwaite**[2], **Anne De Roeck**[1]
[1] Department of Computing, [2] Department of Statistics
The Open University
Milton Keynes, MK7 6AA, UK
{a.sarkar, p.h.garthwaite, a.deroeck}@open.ac.uk

Abstract

This paper proposes a model for term re-occurrence in a text collection based on the gaps between successive occurrences of a term. These gaps are modeled using a mixture of exponential distributions. Parameter estimation is based on a Bayesian framework that allows us to fit a flexible model. The model provides measures of a term's re-occurrence rate and *within-document burstiness*. The model works for all kinds of terms, be it rare content word, medium frequency term or frequent function word. A measure is proposed to account for the term's importance based on its distribution pattern in the corpus.

1 Introduction

Traditionally, Information Retrieval (IR) and Statistical Natural Language Processing (NLP) applications have been based on the "bag of words" model. This model assumes term independence and homogeneity of the text and document under consideration, i.e. the terms in a document are all assumed to be distributed homogeneously. This immediately leads to the Vector Space representation of text. The immense popularity of this model is due to the ease with which mathematical and statistical techniques can be applied to it.

The model assumes that once a term occurs in a document, its overall frequency in the entire document is the only useful measure that associates a term with a document. It does not take into consideration whether the term occurred in the beginning, middle or end of the document. Neither does it consider whether the term occurs many times in close succession or whether it occurs uniformly throughout the document. It also assumes that additional positional information does not provide any extra leverage to the performance of the NLP and IR applications based on it. This assumption has been shown to be wrong in certain applications (Franz, 1997).

Existing models for term distribution are based on the above assumption, so they can merely estimate the term's frequency in a document or a term's topical behavior for a content term. The occurrence of a content word is classified as *topical* or *non-topical* based on whether it occurs once or many times in the document (Katz, 1996). We are not aware of any existing model that makes less stringent assumptions and models the distribution of occurrences of a term.

In this paper we describe a model for term re-occurrence in text based on the gaps between successive occurrences of the term and the position of its first occurrence in a document. The gaps are modeled by a mixture of exponential distributions. Non-occurrence of a term in a document is modeled by the statistical concept of *censoring*, which states that the event of observing a certain term is censored at the end of the document, i.e. the document length. The modeling is done in a Bayesian framework.

The organization of the paper is as follows. In section 2 we discuss existing term distribution models, the issue of burstiness and some other work that demonstrates the failure of the "bag of words" as-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 48–55, Ann Arbor, June 2005.

sumption. In section 3 we describe our mixture model, the issue of censoring and the Bayesian formulation of the model. Section 4 describes the Bayesian estimation theory and methodology. In section 5 we talk about ways of drawing inferences from our model, present parameter estimates on some chosen terms and present case studies for a few selected terms. We discuss our conclusions and suggest directions for future work in section 6.

2 Existing Work

2.1 Models

Previous attempts to model a term's distribution pattern have been based on the Poisson distribution. If the number of occurrences of a term in a document is denoted by k, then the model assumes:

$$p(k) = e^{-\lambda}\frac{\lambda^k}{k!}$$

for $k = 0, 1, 2, \ldots$ Estimates based on this model are good for non-content, non-informative terms, but not for the more informative content terms (Manning and Schütze, 1999).

The two-Poisson model is suggested as a variation of the Poisson distribution (Bookstein and Swanson, 1974; Church and Gale, 1995b). This model assumes that there are two classes of documents associated with a term, one class with a low average number of occurrences and the other with a high average number of occurrences.

$$p(k) = \alpha e^{-\lambda_1}\frac{\lambda_1^k}{k!} + (1-\alpha)e^{-\lambda_2}\frac{\lambda_2^k}{k!},$$

where α and $(1-\alpha)$ denote the probabilities of a document in each of these classes. Often this model under-estimates the probability that a term will occur exactly twice in a document.

2.2 Burstiness

Burstiness is a phenomenon of content words, whereby they are likely to occur again in a text after they have occurred once. Katz (1996) describes *within-document burstiness* as the close proximity of all or some individual instances of a word within a document exhibiting multiple occurrences.

He proposes a model for within-document burstiness with three parameters as:

- the probability that a term occurs in a document at all (document frequency)
- the probability that it will occur a second time in a document given that it has occurred once
- the probability that it will occur another time, given that it has already occurred k times (where k $>$ 1).

The drawbacks of this model are: (a) it cannot handle non-occurrence of a term in a document; (b) the model can handle only content terms, and is not suitable for high frequency function words or medium frequency terms; and (c) the rate of re-occurrence of the term or the length of gaps cannot be accounted for. We overcome these drawbacks in our model.

A measure of burstiness was proposed as a binary value that is based on the magnitude of average-term frequency of the term in the corpus (Kwok, 1996). This measure takes the value 1 (bursty term) if the average-term frequency value is large and 0 otherwise. The measure is too naive and incomplete to account for term burstiness.

2.3 Homogeneity Assumption

The popular "bag of words" assumption for text states that a term's occurrence is uniform and homogeneous throughout. A measure of homogeneity or self-similarity of a corpus can be calculated, by dividing the corpus into two frequency lists based on the term frequency and then calculating the χ^2 statistic between them (Kilgarriff, 1997). Various schemes for dividing the corpus were used (De Roeck et al., 2004a) to detect homogeneity of terms at document level, within-document level and by choosing text chunks of various sizes. Their work revealed that homogeneity increases by nullifying the within document term distribution pattern and homogeneity decreases when chunks of larger size are chosen as it incorporates more document structure in it. Other work based on the same methodology (De Roeck et al., 2004b) reveals that even very frequent function words do not distribute homogeneously over a corpus or document. These (De Roeck et al., 2004a; De Roeck et al., 2004b) provide evidence of the fact that the "bag of words" assumption is invalid. Thus it sets the platform for a model

that defies the independence assumption and considers the term distribution pattern in a document and corpus.

3 Modeling

3.1 Terminology and Notation

We build a single model for a particular term in a given corpus. Let us suppose the term under consideration is x as shown in Figure 1. We describe the notation for a particular document, i in the corpus.

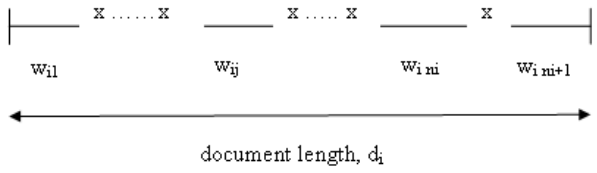


Figure 1: The document structure and the gaps between terms

- d_i denotes the number of words in document i (i.e. the document length).
- n_i denotes the number of occurrences of term x in document i.
- w_{i1} denotes the position of the first occurrence of term x in document i.
- $w_{i2}, \ldots, w_{in_i}$ denotes the successive gaps between occurrences of term x in document i.
- w_{in_i+1} denotes the gap for the next occurrence of x, somewhere after the document ends.
- cen_i is the value at which observation w_{in_i+1} is censored, as explained in section 3.2.2.

3.2 The Model

We suppose we are looking through a document, noting when the term of interest occurs. Our model assumes that the term occurs at some low underlying base rate $1/\lambda_1$ but, after the term has occurred, then the probability of it occurring soon afterwards is increased to some higher rate $1/\lambda_2$. Specifically, the rate of re-occurrence is modeled by a mixture of two exponential distributions. Each of the exponential components is described as follows:

- The exponential component with larger mean (average), $1/\lambda_1$, determines the rate with which the particular term will occur if it has not occurred before or it has not occurred recently.
- The second component with smaller mean (average), $1/\lambda_2$, determines the rate of re-occurrence in a document or text chunk given that it has already occurred recently. This component captures the bursty nature of the term in the text (or document) i.e. the *within-document burstiness*.

The mixture model is described as follows:

$$\phi(w_{ij}) \quad = \quad p\lambda_1 e^{-\lambda_1 w_{ij}} + (1-p)\lambda_2 e^{-\lambda_2 w_{ij}}$$

for $j \in \{2, \ldots, n_i\}$. p and $(1-p)$ denote respectively, the probabilities of membership for the first and the second exponential distribution.

There are a few boundary conditions that the model is expected to handle. We take each of these cases and discuss them briefly:

3.2.1 First occurrence

The model treats the first occurrence of a term differently from the other gaps. The second exponential component measuring burstiness does not feature in it. Hence the distribution is:

$$\phi_1(w_{i1}) \quad = \quad \lambda_1 e^{-\lambda_1 w_{i1}}$$

3.2.2 Censoring

Here we discuss the modeling of two cases that require special attention, corresponding to gaps that have a minimum length but whose actual length is unknown. These cases are:

- The last occurrence of a term in a document.
- The term does not occur in a document at all.

We follow a standard technique from clinical trials, where a patient is observed for a certain amount of time and the observation of the study is expected in that time period (the observation might be the time until death, for example). In some cases it happens that the observation for a patient does not occur in that time period. In such a case it is assumed that the observation would occur at sometime in the future. This is called *censoring* at a certain point.

In our case, we assume the particular term would eventually occur, but the document has ended before it occurs so we do not observe it. In our notation we observe the term n_i times, so the $(n_i+1)^{th}$ time the term occurs is *after* the end of the document. Hence the distribution of w_{in_i+1} is censored at length cen_i. If cen_i is small, so that the n_i^{th} occurrence of the term is near the end of the document, then it is not surprising that w_{in_i+1} is censored. In contrast if cen_i is large, so the n_i^{th} occurrence is far from the end of the document, then either it is surprising that the term did not re-occur, or it suggests the term is rare. The information about the model parameters that is given by the censored occurrence is,

$$Pr(w_{in_i+1} > cen_i) \quad = \quad \int_{cen_i}^{\infty} \phi(x)dx$$

$= pe^{-\lambda_1 cen_i} + (1-p)e^{-\lambda_2 cen_i}$; where,

$$cen_i \quad = \quad d_i - \sum_{j=1}^{n_i} w_{ij}$$

Also when a particular term does not occur in a document, our model assumes that the term would eventually occur had the document continued indefinitely. In this case the first occurrence is censored and censoring takes place at the document length. If a term does not occur in a long document, it suggests the term is rare.

3.3 Bayesian formulation

Our modeling is based on a *Bayesian* approach (Gelman et al., 1995). The Bayesian approach differs from the traditional frequentist approach. In the frequentist approach it is assumed that the parameters of a distribution are constant and the data varies. In the Bayesian approach one can assign distributions to the parameters in a model. We choose non-informative priors, as is common practice in Bayesian applications. So we put,
$p \sim Uniform(0,1)$, and
$\lambda_1 \sim Uniform(0,1)$
To tell the model that λ_2 is the larger of the two λs, we put $\lambda_2 = \lambda_1 + \gamma$, where $\gamma > 0$, and
$\gamma \sim Uniform(0,1)$
Also cen_i depends on the document length d_i and the number of occurrences of the term in that document, n_i. Fitting mixture techniques is tricky and requires special methods. We use data augmentation to make it feasible to fit the model using Gibbs Sampling (section 4.2). For details about this, see Robert (1996) who describes in detail the fitting of mixture models in MCMC methods (section 4.2).

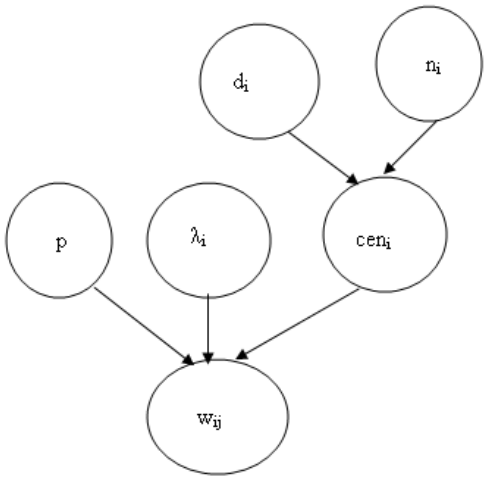


Figure 2: Bayesian dependencies between the parameters

4 Parameter Estimation

4.1 Bayesian Estimation

In the Bayesian approach of parameter estimation, the parameters are uncertain, and it is assumed that they follow some distribution. In our case the parameters and the data are defined as:
$\vec{\Theta} = \{p, \lambda_1, \lambda_2\}$ denote the parameters of the model.
$\vec{W} = \{w_{i1}, \ldots, w_{in_i}, w_{in_i+1}\}$ denotes the data.
Hence based on this we may define the following:

- $f(\vec{\Theta})$ is the **prior distribution** of $\vec{\Theta}$ as assigned in section 3.3. It summarizes everything we know about $\vec{\Theta}$ apart from the data $\vec{W}$.

- $f(\vec{W}|\vec{\Theta})$ is the **likelihood function**. It is our model for the data $\vec{W}$ conditional on the parameters $\vec{\Theta}$. (As well as the observed data, the likelihood also conveys the information given by the censored values)

- $f(\vec{\Theta}|\vec{W})$ is the **posterior distribution** of $\vec{\Theta}$, given $\vec{W}$. It describes our beliefs about the parameters given the information we have.

Deriving the density function for a parameter set $\vec{\Theta}$ after observing data $\vec{W}$, can be achieved by using **Bayes Theorem** as:

$$f(\vec{\Theta}|\vec{W}) = \frac{f(\vec{W}|\vec{\Theta})f(\vec{\Theta})}{f(\vec{W})} \quad (1)$$

where $f(\vec{W})$ is simply a normalizing constant, independent of $\vec{\Theta}$. It can be computed in terms of the likelihood and prior as:

$$f(\vec{W}) = \int f(\vec{W}|\vec{\Theta})f(\vec{\Theta})d\vec{\Theta}$$

Hence equation 1 is reduced to:

$$f(\vec{\Theta}|\vec{W}) \propto f(\vec{W}|\vec{\Theta})f(\vec{\Theta})$$

So, once we have specified the posterior density function $f(\vec{\Theta}|\vec{W})$, we can obtain the estimates of the parameters $\vec{\Theta}$ by simply averaging the values generated by $f(\vec{\Theta}|\vec{W})$.

4.2 Gibbs Sampling

The density function of Θ_i, $f(\Theta_i|\vec{W})$ can be obtained by integrating $f(\vec{\Theta}|\vec{W})$ over the remaining parameters of $\vec{\Theta}$. But in many cases, as in ours, it is impossible to find a closed form solution of $f(\Theta_i)$.

In such cases we may use a simulation process based on random numbers, **Markov Chain Monte Carlo (MCMC)** (Gilks et al., 1996). By generating a large sample of observations from the joint distribution $f(\vec{\Theta}, \vec{W})$, the integrals of the complex distributions can be approximated from the generated data. The values are generated based on the Markov chain assumption, which states that the next generated value only depends on the present value and does not depend on the values previous to it. Based on mild regularity conditions, the chain will gradually *forget* its initial starting point and will eventually converge to a unique *stationary distribution*.

Gibbs Sampling (Gilks et al., 1996) is a popular method used for MCMC analysis. It provides an elegant way for sampling from the joint distributions of multiple variables: sample repeatedly from the distributions of one-dimensional conditionals given the current observations. Initial random values are assigned to each of the parameters. And then these values are updated iteratively based on the joint distribution, until the values settle down and converge to a stationary distribution. The values generated from the start to the point where the chain settles down are discarded and are called the *burn-in* values. The parameter estimates are based on the values generated thereafter.

5 Results

Parameter estimation was carried out using Gibb's Sampling on the WinBUGS software (Spiegelhalter et al., 2003). Values from the first 1000 iteration were discarded as burn-in. It had been observed that in most cases the chain reached the stationary distribution well within 1000 iterations. A further 5000 iterations were run to obtain the parameter estimates.

5.1 Interpretation of Parameters

The parameters of the model can be interpreted in the following manner:

- $\widetilde{\lambda_1} = 1/\lambda_1$ is the mean of an exponential distribution with parameter λ_1. $\widetilde{\lambda_1}$ measures the rate at which this term is expected in a running text corpus. $\widetilde{\lambda_1}$ determines the rarity of a term in a corpus, as it is the average gap at which the term occurs if it has not occurred recently. Thus, a large value of $\widetilde{\lambda_1}$ tells us that the term is very rare in the corpus and vice-versa.

- Similarly, $\widetilde{\lambda_2}$ measures the *within-document burstiness*, i.e. the rate of occurrence of a term given that it has occurred recently. It measures the term re-occurrence rate in a burst within a document. Small values of $\widetilde{\lambda_2}$ indicate the bursty nature of the term.

- $\widetilde{p}$ and $1 - \widetilde{p}$ denote, respectively, the probabilities of the term occurring with rate $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$ in the entire corpus.

Table 1 presents some heuristics for drawing inference based on the values of the parameter estimates.

5.2 Data

We choose for evaluation, terms from the *Associated Press (AP)* newswire articles, as this is a standard corpus for language research. We picked terms which had been used previously in the literature (Church and Gale, 1995a; Church, 2000; Manning

	$\widetilde{\lambda_1}$ small	$\widetilde{\lambda_1}$ large
$\widetilde{\lambda_2}$ small	frequently occurring and common function word	topical content word occurring in bursts
$\widetilde{\lambda_2}$ large	comparatively frequent but well-spaced function word	infrequent and scattered function word

Table 1: Heuristics for inference, based on the parameter estimates.

and Schütze, 1999; Umemura and Church, 2000) with respect to modeling different distribution, so as to present a comparative picture. For building the model we randomly selected 1% of the documents from the corpus, as the software (Spiegelhalter et al., 2003) we used is Windows PC based and could not handle enormous volume of data with our available hardware resources. As stated earlier, our model can handle both frequent function terms and rare content terms. We chose terms suitable for demonstrating this. We also used some medium frequency terms to demonstrate their characteristics.

5.3 Parameter estimates

Table 2 shows the parameter estimates for the chosen terms. The table does not show the values of $1-\widetilde{p}$ as they can be obtained from the value of $\widetilde{p}$. It has been observed that the value $\widetilde{\lambda_1}/\widetilde{\lambda_2}$ is a good indicator of the nature of terms, hence the rows in the table containing terms are sorted on the basis of that value. The table is divided into three parts. The top part contains very frequent (function) words. The second part contains terms in the medium frequency range. And the bottom part contains rarely occurring and content terms.

5.4 Discussion

The top part of the table consists of the very frequently occurring function words occurring frequently throughout the corpus. These statements are supported by the low values of $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$. These values are quite close, indicating that the occurrence of these terms shows low burstiness in a running text chunk. This supports our heuristics about the value of $\widetilde{\lambda_1}/\widetilde{\lambda_2}$, which is small for such terms. Moderate, not very high values of $\widetilde{p}$ also support this statement, as the term is then quite likely to be generated from either of the exponential distributions (*the* has high value of $\widetilde{p}$, but since the values of λ are so close, it doesn't really matter which distribution generated the observation). We observe comparatively larger values of $\widetilde{\lambda_1}$ for terms like *yet*, *follows* and *except* since they have some dependence on the document topic. One may claim that these are some outliers having large values of both $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$. The large value of $\widetilde{\lambda_1}$ can be explained, as these terms are rarely occurring function words in the corpus. They do not occur in bursts and their occurrences are scattered, so values of $\widetilde{\lambda_2}$ are also large (Table 1). Interestingly, based on our heuristics these large values nullify each other to obtain a small value of $\widetilde{\lambda_1}/\widetilde{\lambda_2}$. But since these cases are exceptional, they find their place on the boundary region of the division.

Term	$\widetilde{p}$	$\widetilde{\lambda_1}$	$\widetilde{\lambda_2}$	$\widetilde{\lambda_1}/\widetilde{\lambda_2}$
the	0.82	16.54	16.08	1.03
and	0.46	46.86	45.19	1.04
of	0.58	38.85	37.22	1.04
except	0.67	21551.72	8496.18	2.54
follows	0.56	80000.00	30330.60	2.64
yet	0.51	10789.81	3846.15	2.81
he	0.51	296.12	48.22	6.14
said	0.03	895.26	69.06	12.96
government	0.60	1975.50	134.34	14.71
somewhat	0.84	75244.54	4349.72	17.30
federal	0.84	2334.27	102.57	22.76
here	0.94	3442.34	110.63	31.12
she	0.73	1696.35	41.41	40.97
george	0.88	17379.21	323.73	53.68
bush	0.71	3844.68	53.48	71.90
soviet	0.71	4496.40	59.74	75.27
kennedy	0.78	14641.29	99.11	147.73
church	0.92	11291.78	70.13	161.02
book	0.92	17143.84	79.68	215.16
vietnam	0.92	32701.11	97.66	334.86
boycott	0.98	105630.08	110.56	955.42
noriega	0.91	86281.28	56.88	1516.82

Table 2: Parameter estimates of the model for some selected terms, sorted by the $\widetilde{\lambda_1}/\widetilde{\lambda_2}$ value

The second part of the table contains mostly *non-topical content terms* as defined in the literature (Katz, 1996). They do not describe the main topic of the document, but some useful aspects of the document or a nearby topical term. Special attention may be given to the term *george*, which describes the topical term *bush*. In a document about *George Bush*, the complete name is mentioned possibly only once in the beginning and further references to it are made using the word *bush*, leading to *bush* being as-

signed as a topical term, but not *george*. The term *government* in the group refers to some newswire article about some government in any state or any country, future references to which are made using this term. Similarly the term *federal* is used to make future references to the *US Government*. As the words *federal* and *government* are used frequently for referencing, they exhibit comparatively small values of $\widetilde{\lambda_2}$. We were surprised by the occurrence of terms like *said, here* and *she* in the second group, as they are commonly considered as function words. Closer examination revealed the details. *Said* has some dependence on the document genre, with respect to the content and reporting style. The data were based on newswire articles about important people and events. It is true, though unfortunate, that the majority of such people are male, hence there are more articles about men than women (*he* occurs $757,301$ times in $163,884$ documents as the 13^{th} most frequent term in the corpus, whereas *she* occurs $164,030$ times in $48,794$ documents as the 70^{th} frequent term). This explains why *he* has a smaller value of $\widetilde{\lambda_1}$ than *she*. But the $\widetilde{\lambda_2}$ values for both of them are quite close, showing that they have similar usage pattern. Again, newswire articles are mostly about people and events, and rarely about some location, referenced by the term *here*. This explains the large value of $\widetilde{\lambda_1}$ for *here*. Again, because of its usage for referencing, it re-occurs frequently while describing a particular location, leading to a small value of $\widetilde{\lambda_2}$. Possibly, in a collection of "travel documents", *here* will have a smaller value of $\widetilde{\lambda_1}$ and thus occur higher up in the list, which would allow the model to be used for characterizing genre.

Terms in the third part, as expected, are *topical content terms*. An occurrence of such a term defines the topic or the main content word of the document or the text chunk under consideration. These terms are rare in the entire corpus, and only appear in documents that are about this term, resulting in very high values of $\widetilde{\lambda_1}$. Also low values of $\widetilde{\lambda_2}$ for these terms mean that repeat occurrences within the same document are quite frequent; the characteristic expected from a topical content term. Because of these characteristics, based on our heuristics these terms have very high values of $\widetilde{\lambda_1}/\widetilde{\lambda_2}$, and hence are considered the most informative terms in the corpus.

5.5 Case Studies

Here we study selected terms based on our model. These terms have been studied before by other researchers. We study these terms to compare our findings with previous work and also demonstrate the range of inferences that may be derived from our model.

5.5.1 somewhat *vrs* boycott

These terms occur an approximately equal number of times in the AP corpus, and *inverse document frequency* was used to distinguish between them (Church and Gale, 1995a). Our model also gives approximately similar rates of occurrence ($\widetilde{\lambda_1}$) for these two terms as shown in Table 2. But the reoccurrence rate, $\widetilde{\lambda_2}$, is 110.56 for *boycott*, which is very small in comparison with the value of 4349.72 for *somewhat*. Hence based on this, our model assigns *somewhat* as a rare function word occurring in a scattered manner over the entire corpus. Whereas *boycott* is assigned as a topical content word, as it should be.

5.5.2 follows *vrs* soviet

These terms were studied in connection with fitting Poisson distributions to their term distribution (Manning and Schütze, 1999), and hence determining their characteristics[1]. In our model, *follows* has large values of both $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$ (Table 2), so that it has the characteristics of a rare function word. But *soviet* has a large $\widetilde{\lambda_1}$ value and a very small $\widetilde{\lambda_2}$ value, so that it has the characteristics of a topical content word. So the findings from our model agree with the original work.

5.5.3 kennedy *vrs* except

Both these terms have nearly equal *inverse document frequency* for the AP corpus (Church, 2000; Umemura and Church, 2000) and will be assigned equal weight. They used a method (Kwok, 1996) based on average-term frequency to determine the nature of the term. According to our model, the $\widetilde{\lambda_2}$ value of *kennedy* is very small as compared to that for *except*. Hence using the $\widetilde{\lambda_1}/\widetilde{\lambda_2}$ measure, we can correctly identify *kennedy* as a topical content term

[1]The original study was based on the *New York Times*, ours on the *Associated Press* corpus

and *except* as an infrequent function word. This is in agreement with the findings of the original analysis.

5.5.4 noriega *and* said

These terms were studied in the context of an adaptive language model to demonstrate the fact that the probability of a repeat occurrence of a term in a document defies the "bag of words" independence assumption (Church, 2000). The deviation from independence is greater for content terms like *noriega* as compared to general terms like *said*. This can be explained in the context of our model as *said* has small values of $\widetilde{\lambda_1}$ and $\widetilde{\lambda_2}$, and their values are quite close to each other (as compared to other terms, see Table 2). Hence *said* is distributed more evenly in the corpus than *noriega*. Therefore, *noriega* defies the independence assumption to a much greater extent than *said*. Hence their findings (Church, 2000) are well explained by our model.

6 Conclusion

In this paper we present a model for term reoccurrence in text based on gaps between successive occurrences of a term in a document. Parameter estimates based on this model reveal various characteristics of term use in a collection. The model can differentiate a term's dependence on genre and collection and we intend to investigate use of the model for purposes like genre detection, corpus profiling, authorship attribution, text classification, etc. The proposed measure of $\widetilde{\lambda_1}/\widetilde{\lambda_2}$ can be appropriately adopted as a means of feature selection that takes into account the term's occurrence pattern in a corpus. We can capture both within-document burstiness and rate of occurrence of a term in a single model.

References

A. Bookstein and D.R Swanson. 1974. Probabilistic models for automatic indexing. *Journal of the American Society for Information Science*, 25:312–318.

K. Church and W. Gale. 1995a. Inverse document frequency (idf): A measure of deviation from poisson. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 121–130.

K. Church and W. Gale. 1995b. Poisson mixtures. *Natural Language Engineering*, 1(2):163–190.

K. Church. 2000. Empirical estimates of adaptation: The chance of two noriega's is closer to p/2 than p^2. In *COLING*, pages 173–179.

Anne De Roeck, Avik Sarkar, and Paul H Garthwaite. 2004a. Defeating the homogeneity assumption. In *Proceedings of 7th International Conference on the Statistical Analysis of Textual Data (JADT)*, pages 282–294.

Anne De Roeck, Avik Sarkar, and Paul H Garthwaite. 2004b. Frequent term distribution measures for dataset profiling. In *Proceedings of the 4th International conference of Language Resources and Evaluation (LREC)*, pages 1647–1650.

Alexander Franz. 1997. Independence assumptions considered harmful. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 182–189.

A. Gelman, J. Carlin, H.S. Stern, and D.B. Rubin. 1995. *Bayesian Data Analysis*. Chapman and Hall, London, UK.

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. 1996. *Markov Chain Monte Carlo in Practice*. Interdisciplinary Statistics Series. Chapman and Hall, London, UK.

Slava M. Katz. 1996. Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, 2(1):15–60.

A Kilgarriff. 1997. Using word frequency lists to measure corpus homogeneity and similarity between corpora. In *Proceedings of ACL-SIGDAT Workshop on very large corpora*, Hong Kong.

K. L. Kwok. 1996. A new method of weighting query terms for ad-hoc retrieval. In *SIGIR*, pages 187–195.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Christian. P. Robert. 1996. Mixtures of distributions: inference and estimation. In W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors, *Markov Chain Monte Carlo in Practice*, pages 441–464.

D.J. Spiegelhalter, A. Thomas, N. G. Best, and D. Lunn. 2003. Winbugs: Windows version of bayesian inference using gibbs sampling, version 1.4.

K. Umemura and K. Church. 2000. Empirical term weighting and expansion frequency. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 117–123.

Domain Kernels for Text Categorization

Alfio Gliozzo and **Carlo Strapparava**
ITC-Irst
via Sommarive, I-38050, Trento, ITALY
{gliozzo,strappa}@itc.it

Abstract

In this paper we propose and evaluate a technique to perform semi-supervised learning for Text Categorization. In particular we defined a kernel function, namely the Domain Kernel, that allowed us to plug "external knowledge" into the supervised learning process. External knowledge is acquired from unlabeled data in a totally unsupervised way, and it is represented by means of Domain Models.

We evaluated the Domain Kernel in two standard benchmarks for Text Categorization with good results, and we compared its performance with a kernel function that exploits a standard bag-of-words feature representation. The learning curves show that the Domain Kernel allows us to reduce drastically the amount of training data required for learning.

1 Introduction

Text Categorization (TC) deals with the problem of assigning a set of category labels to documents. Categories are usually defined according to a variety of topics (e.g. SPORT vs. POLITICS) and a set of hand tagged examples is provided for training. In the state-of-the-art TC settings supervised classifiers are used for learning and texts are represented by means of *bag-of-words*.

Even if, in principle, supervised approaches reach the best performance in many Natural Language Processing (NLP) tasks, in practice it is not always easy to apply them to concrete applicative settings. In fact, supervised systems for TC require to be trained a large amount of hand tagged texts. This situation is usually feasible only when there is someone (e.g. a big company) that can easily provide already classified documents to train the system.

In most of the cases this scenario is quite unpractical, if not infeasible. An example is the task of categorizing personal documents, in which the categories can be modified according to the user's interests: new categories are often introduced and, possibly, the available labeled training for them is very limited.

In the NLP literature the problem of providing large amounts of manually annotated data is known as the Knowledge Acquisition Bottleneck. Current research in supervised approaches to NLP often deals with defining methodologies and algorithms to reduce the amount of human effort required for collecting labeled examples.

A promising direction to solve this problem is to provide unlabeled data together with labeled texts to help supervision. In the Machine Learning literature this learning schema has been called *semi-supervised learning*. It has been applied to the TC problem using different techniques: co-training (Blum and Mitchell, 1998), EM-algorithm (Nigam et al., 2000), Transduptive SVM (Joachims, 1999b) and Latent Semantic Indexing (Zelikovitz and Hirsh, 2001).

In this paper we propose a novel technique to perform semi-supervised learning for TC. The underlying idea behind our approach is that lexical co-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 56–63, Ann Arbor, June 2005.

herence (i.e. co-occurence in texts of semantically related terms) (Magnini et al., 2002) is an inherent property of corpora, and it can be exploited to help a supervised classifier to build a better categorization hypothesis, even if the amount of labeled training data provided for learning is very low.

Our proposal consists of defining a Domain Kernel and exploiting it inside a Support Vector Machine (SVM) classification framework for TC (Joachims, 2002). The Domain Kernel relies on the notion of Domain Model, which is a shallow representation for lexical ambiguity and variability. Domain Models can be acquired in an unsupervised way from unlabeled data, and then exploited to define a Domain Kernel (i.e. a generalized similarity function among documents)[1].

We evaluated the Domain Kernel in two standard benchmarks for TC (i.e. Reuters and 20Newsgroups), and we compared its performance with a kernel function that exploits a more standard Bag-of-Words (BoW) feature representation. The use of the Domain Kernel got a significant improvement in the learning curves of both tasks. In particular, there is a notable increment of the recall, especially with few learning examples. In addition, F1 measure increases by 2.8 points in the Reuters task at full learning, achieving the state-of-the-art results.

The paper is structured as follows. Section 2 introduces the notion of Domain Model and describes an automatic acquisition technique based on Latent Semantic Analysis (LSA). In Section 3 we illustrate the SVM approach to TC, and we define a Domain Kernel that exploits Domain Models to estimate similarity among documents. In Section 4 the performance of the Domain Kernel are compared with a standard bag-of-words feature representation, showing the improvements in the learning curves. Section 5 describes the previous attempts to exploit semi-supervised learning for TC, while section 6 concludes the paper and proposes some directions for future research.

[1]The idea of exploiting a Domain Kernel to help a supervised classification framework, has been profitably used also in other NLP tasks such as word sense disambiguation (see for example (Strapparava et al., 2004)).

2 Domain Models

The simplest methodology to estimate the similarity among the topics of two texts is to represent them by means of vectors in the Vector Space Model (VSM), and to exploit the cosine similarity. More formally, let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be a corpus, let $V = \{w_1, w_2, \ldots, w_k\}$ be its vocabulary, let $\mathbf{T}$ be the $k \times n$ term-by-document matrix representing $\mathcal{T}$, such that $\mathbf{t_{i,j}}$ is the frequency of word w_i into the text t_j. The VSM is a k-dimensional space $\mathbf{R}^k$, in which the text $t_j \in \mathcal{T}$ is represented by means of the vector $\vec{t_j}$ such that the i^{th} component of $\vec{t_j}$ is $\mathbf{t_{i,j}}$. The similarity among two texts in the VSM is estimated by computing the cosine.

However this approach does not deal well with lexical variability and ambiguity. For example the two sentences "*he is affected by AIDS*" and "*HIV is a virus*" do not have any words in common. In the VSM their similarity is zero because they have orthogonal vectors, even if the concepts they express are very closely related. On the other hand, the similarity between the two sentences "*the laptop has been infected by a **virus***" and "*HIV is a **virus***" would turn out very high, due to the ambiguity of the word `virus`.

To overcome this problem we introduce the notion of *Domain Model* (DM), and we show how to use it in order to define a *domain VSM*, in which texts and terms are represented in a uniform way.

A Domain Model is composed by soft clusters of terms. Each cluster represents a semantic domain (Gliozzo et al., 2004), i.e. a set of terms that often co-occur in texts having similar topics. A Domain Model is represented by a $k \times k'$ rectangular matrix $\mathbf{D}$, containing the degree of association among terms and domains, as illustrated in Table 1.

	MEDICINE	COMPUTER_SCIENCE
HIV	1	0
AIDS	1	0
virus	0.5	0.5
laptop	0	1

Table 1: Example of Domain Matrix

Domain Models can be used to describe lexical ambiguity and variability. Lexical ambiguity is rep-

resented by associating one term to more than one domain, while variability is represented by associating different terms to the same domain. For example the term `virus` is associated to both the domain COMPUTER_SCIENCE and the domain MEDICINE (ambiguity) while the domain MEDICINE is associated to both the terms `AIDS` and `HIV` (variability).

More formally, let $\mathcal{D} = \{D_1, D_2, ..., D_{k'}\}$ be a set of domains, such that $k' \ll k$. A Domain Model is fully defined by a $k \times k'$ *domain matrix* $\mathbf{D}$ representing in each cell $\mathbf{d_{i,z}}$ the *domain relevance* of term w_i with respect to the domain D_z. The domain matrix $\mathbf{D}$ is used to define a function $\mathcal{D} : \mathbf{R}^k \rightarrow \mathbf{R}^{k'}$, that maps the vectors $\vec{t_j}$, expressed into the classical VSM, into the vectors $\vec{t'_j}$ in the domain VSM. $\mathcal{D}$ is defined by[2]

$$\mathcal{D}(\vec{t_j}) = \vec{t_j}(\mathbf{I^{IDF}D}) = \vec{t'_j} \quad (1)$$

where $\mathbf{I^{IDF}}$ is a diagonal matrix such that $i^{IDF}_{i,i} = IDF(w_i)$, $\vec{t_j}$ is represented as a row vector, and $IDF(w_i)$ is the *Inverse Document Frequency* of w_i.

Vectors in the domain VSM are called Domain Vectors. Domain Vectors for texts are estimated by exploiting formula 1, while the Domain Vector $\vec{w'_i}$, corresponding to the word $w_i \in V$, is the i^{th} row of the domain matrix $\mathbf{D}$. To be a valid domain matrix such vectors should be normalized (i.e. $\langle \vec{w'_i}, \vec{w'_i} \rangle = 1$).

In the Domain VSM the similarity among Domain Vectors is estimated by taking into account second order relations among terms. For example the similarity of the two sentences "*He is affected by AIDS*" and "*HIV is a virus*" is very high, because the terms `AIDS`, `HIV` and `virus` are highly associated to the domain MEDICINE.

In this work we propose the use of Latent Semantic Analysis (LSA) (Deerwester et al., 1990) to induce Domain Models from corpora. LSA is an unsupervised technique for estimating the similarity among texts and terms in a corpus. LSA is performed by means of a Singular Value Decomposition (SVD) of the term-by-document matrix $\mathbf{T}$ describing the corpus. The SVD algorithm can be exploited to acquire a domain matrix $\mathbf{D}$ from a large corpus $\mathcal{T}$ in a totally unsupervised way. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrixes $\mathbf{T} \simeq \mathbf{V\Sigma_{k'}U}^T$ where $\mathbf{\Sigma_{k'}}$ is the diagonal $k \times k$ matrix containing the highest $k' \ll k$ eigenvalues of $\mathbf{T}$, and all the remaining elements set to 0. The parameter k' is the dimensionality of the Domain VSM and can be fixed in advance[3]. Under this setting we define the domain matrix $\mathbf{D_{LSA}}$[4] as

$$\mathbf{D_{LSA}} = \mathbf{I^N V}\sqrt{\mathbf{\Sigma_{k'}}} \quad (2)$$

where $\mathbf{I^N}$ is a diagonal matrix such that $\mathbf{i^N_{i,i}} = \frac{1}{\sqrt{\langle \vec{w'_i}, \vec{w'_i} \rangle}}$, $\vec{w'_i}$ is the i^{th} row of the matrix $\mathbf{V}\sqrt{\mathbf{\Sigma_{k'}}}$.

3 The Domain Kernel

Kernel Methods are the state-of-the-art supervised framework for learning, and they have been successfully adopted to approach the TC task (Joachims, 1999a).

The basic idea behind kernel methods is to embed the data into a suitable feature space $\mathcal{F}$ via a mapping function $\phi : \mathcal{X} \rightarrow \mathcal{F}$, and then use a linear algorithm for discovering nonlinear patterns. Kernel methods allow us to build a modular system, as the kernel function acts as an interface between the data and the learning algorithm. Thus the kernel function becomes the only domain specific module of the system, while the learning algorithm is a general purpose component. Potentially a kernel function can work with any kernel-based algorithm, such as for example SVM.

During the learning phase SVMs assign a weight $\lambda_i \geq 0$ to any example $x_i \in X$. All the labeled instances x_i such that $\lambda_i > 0$ are called *support vectors*. The support vectors lie close to the best separating hyper-plane between positive and negative examples. New examples are then assigned to the class of its closest support vectors, according to equation 3.

[2] In (Wong et al., 1985) a similar schema is adopted to define a Generalized Vector Space Model, of which the Domain VSM is a particular instance.

[3] It is not clear how to choose the right dimensionality. In our experiments we used 400 dimensions.

[4] When $\mathbf{D_{LSA}}$ is substituted in Equation 1 the Domain VSM is equivalent to a Latent Semantic Space (Deerwester et al., 1990). The only difference in our formulation is that the vectors representing the terms in the Domain VSM are normalized by the matrix $\mathbf{I^N}$, and then rescaled, according to their IDF value, by matrix $\mathbf{I^{IDF}}$. Note the analogy with the *tf idf* term weighting schema (Salton and McGill, 1983), widely adopted in Information Retrieval.

$$f(x) = \sum_{i=1}^{n} \lambda_i K(x_i, x) + \lambda_0 \quad (3)$$

The kernel function K returns the similarity between two instances in the input space X, and can be designed in order to capture the relevant aspects to estimate similarity, just by taking care of satisfying set of formal requirements, as described in (Schölkopf and Smola, 2001).

In this paper we define the Domain Kernel and we apply it to TC tasks. The Domain Kernel, denoted by K_D, can be exploited to estimate the topic similarity among two texts while taking into account the external knowledge provided by a Domain Model (see section 2). It is a variation of the Latent Semantic Kernel (Shawe-Taylor and Cristianini, 2004), in which a Domain Model is exploited to define an explicit mapping $\mathcal{D} : \mathbf{R}^k \rightarrow \mathbf{R}^{k'}$ from the classical VSM into the domain VSM. The Domain Kernel is defined by

$$K_D(t_i, t_j) = \frac{\langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle}{\sqrt{\langle \mathcal{D}(t_j), \mathcal{D}(t_j) \rangle \langle \mathcal{D}(t_i), \mathcal{D}(t_i) \rangle}} \quad (4)$$

where $\mathcal{D}$ is the Domain Mapping defined in equation 1. To be fully defined, the Domain Kernel requires a Domain Matrix $\mathbf{D}$. In principle, $\mathbf{D}$ can be acquired from any corpora by exploiting any (soft) term clustering algorithm. Anyway, we belive that adequate Domain Models for particular tasks can be better acquired from collections of documents from the same source. For this reason, for the experiments reported in this paper, we acquired the matrix $\mathbf{D_{LSA}}$, defined by equation 2, using the whole (unlabeled) training corpora available for each task, so tuning the Domain Model on the particular task in which it will be applied.

A more traditional approach to measure topic similarity among text consists of extracting BoW features and to compare them in a vector space. The BoW kernel, denoted by K_{BoW}, is a particular case of the Domain Kernel, in which $\mathbf{D} = \mathbf{I}$, and $\mathbf{I}$ is the identity matrix. The BoW Kernel does not require a Domain Model, so we can consider this setting as "purely" supervised, in which no external knowledge source is provided.

4 Evaluation

We compared the performance of both K_D and K_{BoW} on two standard TC benchmarks. In subsection 4.1 we describe the evaluation tasks and the preprocessing steps, in 4.2 we describe some algorithmic details of the TC system adopted. Finally in subsection 4.3 we compare the learning curves of K_D and K_{BoW}.

4.1 Text Categorization tasks

For the experiments reported in this paper, we selected two evaluation benchmarks typically used in the TC literature (Sebastiani, 2002): the *20newsgroups* and the *Reuters* corpora. In both the data sets we tagged the texts for part of speech and we considered only the noun, verb, adjective, and adverb parts of speech, representing them by vectors containing the frequencies of each disambiguated lemma. The only feature selection step we performed was to remove all the closed-class words from the document index.

20newsgroups. The *20Newsgroups* data set[5] is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. This collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. Some of the newsgroups are very closely related to each other (e.g. `comp.sys.ibm.pc.hardware` / `comp.sys.mac.hardware`), while others are highly unrelated (e.g. `misc.forsale` / `soc.religion.christian`). We removed cross-posts (duplicates), newsgroup-identifying headers (i.e. Xref, Newsgroups, Path, Followup-To, Date), and empty documents from the original corpus, so to obtain 18,941 documents. Then we randomly divided it into training (80%) and test (20%) sets, containing respectively 15,153 and 3,788 documents.

Reuters. We used the *Reuters-21578* collection[6], and we splitted it into training and test

[5] Available at `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`.

[6] Available at `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`.

partitions according to the standard *ModApté* split. It includes 12,902 documents for 90 categories, with a fixed splitting between training and test data. We conducted our experiments by considering only the 10 most frequent categories, i.e. `Earn`, `Acquisition`, `Money-fx`, `Grain`, `Crude`, `Trade`, `Interest`, `Ship`, `Wheat` and `Corn`, and we included in our dataset all the non empty documents labeled with at least one of those categories. Thus the final dataset includes 9295 document, of which 6680 are included in the training partition, and 2615 are in the test set.

4.2 Implementation details

As a supervised learning device, we used the SVM implementation described in (Joachims, 1999a). The Domain Kernel is implemented by defining an explicit feature mapping according to formula 1, and by normalizing each vector to obtain vectors of unitary length. All the experiments have been performed on the standard parameter settings, using a linear kernel.

We acquired a different Domain Model for each corpus by performing the SVD processes on the term-by-document matrices representing the whole training partitions, and we considered only the first 400 domains (i.e. $k' = 400$)[7].

As far as the *Reuters* task is concerned, the TC problem has been approached as a set of binary filtering problems, allowing the TC system to provide more than one category label to each document. For the *20newsgroups* task, we implemented a one-versus-all classification schema, in order to assign a single category to each news.

4.3 Domain Kernel versus BoW Kernel

Figure 1 and Figure 2 report the learning curves for both K_D and K_{BoW}, evaluated respectively on the *Reuters* and the *20newgroups* task. Results clearly show that K_D always outperforms K_{BoW}, especially when very limited amount of labeled data is provided for learning.

[7]To perform the SVD operation we adopted LIBSVDC, an optimized package for sparse matrix that allows to perform this step in few minutes even for large corpora. It can be downloaded from `http://tedlab.mit.edu/~dr/SVDLIBC/`.

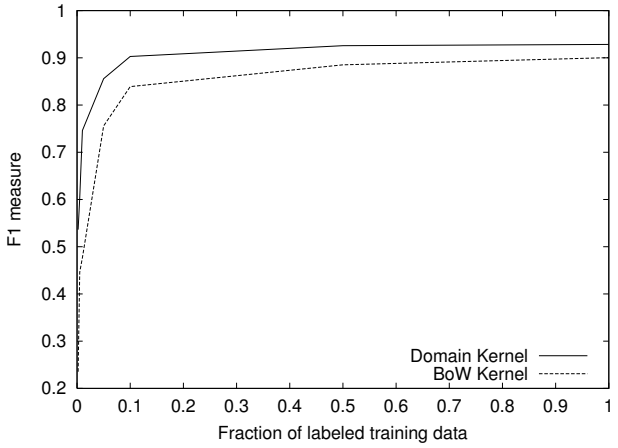


Figure 1: Micro-F1 learning curves for *Reuters*

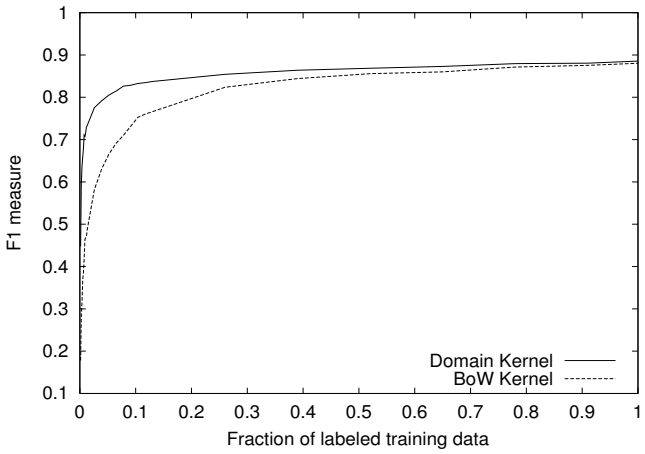


Figure 2: Micro-F1 learning curves for *20newsgroups*

Table 2 compares the performances of the two kernels at full learning. K_D achieves a better micro-F1 than K_{BoW} in both tasks. The improvement is particularly significant in the *Reuters* task (+ 2.8 %).

Tables 3 shows the number of labeled examples required by K_D and K_{BoW} to achieve the same micro-F1 in the *Reuters* task. K_D requires only 146 examples to obtain a micro-F1 of 0.84, while K_{BoW} requires 1380 examples to achieve the same performance. In the same task, K_D surpass the performance of K_{BoW} at full learning using only the 10% of the labeled data. The last column of the table shows clearly that K_D requires 90% less labeled data than K_{BoW} to achieve the same performances.

A similar behavior is reported in Table 4 for the

F1	*Domain Kernel*	*Bow Kernel*
Reuters	**0.928**	0.900
20newsgroups	**0.886**	0.880

Table 2: Micro-F1 with full learning

F1	*Domain Kernel*	*Bow Kernel*	*Ratio*
.54	**14**	267	5%
.84	**146**	1380	10%
.90	**668**	6680	10%

Table 3: Number of training examples needed by K_D and K_{BoW} to reach the same micro-F1 on the *Reuters* task

20newsgroups task. It is important to notice that the number of labeled documents is higher in this corpus than in the previous one. The benefits of using Domain Models are then less evident at full learning, even if they are significant when very few labeled data are provided.

Figures 3 and 4 report a more detailed analysis by comparing the micro-precision and micro-recall learning curves of both kernels in the *Reuters* task[8]. It is clear from the graphs that the main contribute of K_D is about increasing recall, while precision is similar in both cases[9]. This last result confirms our hypothesis that the information provided by the Domain Models allows the system to generalize in a more effective way over the training examples, allowing to estimate the similarity among texts even if they have just few words in common.

Finally, K_D achieves the state-of-the-art in the *Reuters* task, as reported in section 5.

5 Related Works

To our knowledge, the first attempt to apply the semi-supervised learning schema to TC has been reported in (Blum and Mitchell, 1998). Their co-training algorithm was able to reduce significantly the error rate, if compared to a strictly supervised classifier.

(Nigam et al., 2000) adopted an Expectation Maximization (EM) schema to deal with the same problem, evaluating extensively their approach on several datasets. They compared their algorithm with a standard probabilistic approach to TC, reporting substantial improvements in the learning curve.

[8]For the 20-newsgroups task both micro-precision and micro-recall are equal to micro-F1 because a single category label has been assigned to every instance.

[9]It is worth noting that K_D gets a F1 measure of 0.54 (Precision/Recall of 0.93/0.38) using just 14 training examples, suggesting that it can be profitably exploited for a bootstrapping process.

F1	*Domain Kernel*	*Bow Kernel*	*Ratio*
.50	**30**	500	6%
.70	**98**	1182	8%
.85	**2272**	7879	29%

Table 4: Number of training examples needed by K_D and K_{BoW} to reach the same micro-F1 on the *20newsgroups* task

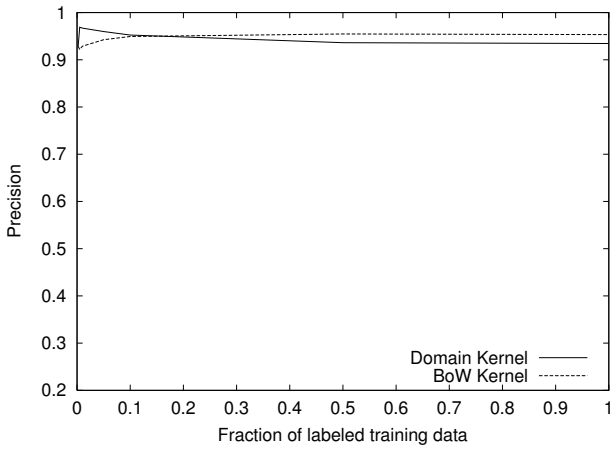


Figure 3: Learning curves for *Reuters* (Precision)

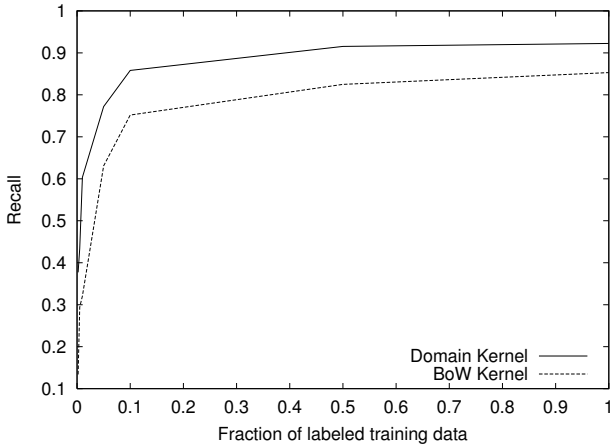


Figure 4: Learning curves for *Reuters* (Recall)

A similar evaluation is also reported in (Joachims, 1999b), where a transduptive SVM is compared to a state-of-the-art TC classifier based on SVM. The semi-supervised approach obtained better results than the standard with few learning data, while at full learning results seem to converge.

(Bekkerman et al., 2002) adopted a SVM classifier in which texts have been represented by their associations to a set of Distributional Word Clusters. Even if this approach is very similar to ours, it is not a semi-supervised learning schema, because authors did not exploit any additional unlabeled data to induce word clusters.

In (Zelikovitz and Hirsh, 2001) background knowledge (i.e. the unlabeled data) is exploited together with labeled data to estimate document similarity in a Latent Semantic Space (Deerwester et al., 1990). Their approach differs from the one proposed in this paper because a different categorization algorithm has been adopted. Authors compared their algorithm with an EM schema (Nigam et al., 2000) on the same dataset, reporting better results only with very few labeled data, while EM performs better with more training.

All the semi-supervised approaches in the literature reports better results than strictly supervised ones with few learning, while with more data the learning curves tend to converge.

A comparative evaluation among semi-supervised TC algorithms is quite difficult, because the used data sets, the preprocessing steps and the splitting partitions adopted affect sensibly the final results. Anyway, we reported the best F1 measure on the Reuters corpus: to our knowledge, the state-of-the-art on the 10 top most frequent categories of the ModApte split at full learning is F1 92.0 (Bekkerman et al., 2002) while we obtained 92.8. It is important to notice here that this results has been obtained thanks to the improvements of the Domain Kernel. In addition, on the *20newsgroups* task, our methods requires about 100 documents (i.e. five documents per category) to achieve 70% F1, while both EM (Nigam et al., 2000) and LSI (Zelikovitz and Hirsh, 2001) requires more than 400 to achieve the same performance.

6 Conclusion and Future Works

In this paper a novel technique to perform semi-supervised learning for TC has been proposed and evaluated. We defined a Domain Kernel that allows us to improve the similarity estimation among documents by exploiting Domain Models. Domain Models are acquired from large collections of non annotated texts in a totally unsupervised way.

An extensive evaluation on two standard benchmarks shows that the Domain Kernel allows us to reduce drastically the amount of training data required for learning. In particular the recall increases sensibly, while preserving a very good accuracy. We explained this phenomenon by showing that the similarity scores evaluated by the Domain Kernel takes into account both variability and ambiguity, being able to estimate similarity even among texts that do not have any word in common.

As future work, we plan to apply our semi-supervised learning method to some concrete applicative scenarios, such as user modeling and categorization of personal documents in mail clients. In addition, we are going deeper in the direction of semi-supervised learning, by acquiring more complex structures than clusters (e.g. synonymy, hyperonymy) to represent domain models. Furthermore, we are working to adapt the general framework provided by the Domain Models to a multilingual scenario, in order to apply the Domain Kernel to a Cross Language TC task.

Acknowledgments

This work has been partially supported by the ONTOTEXT (From Text to Knowledge for the Semantic Web) project, funded by the Autonomous Province of Trento under the FUP-2004 research program.

References

R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. 2002. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 1:1183–1208.

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers.*

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*.

A. Gliozzo, C. Strapparava, and I. Dagan. 2004. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language*, 18:275–299.

T. Joachims. 1999a. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in kernel methods: support vector learning*, chapter 11, pages 169 – 184. MIT Press, Cambridge, MA, USA.

T. Joachims. 1999b. Transductive inference for text classification using support vector machines. In *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann Publishers, San Francisco, US.

T. Joachims. 2002. *Learning to Classify Text using Support Vector Machines*. Kluwer Academic Publishers.

B. Magnini, C. Strapparava, G. Pezzulo, and A. Gliozzo. 2002. The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(4):359–373.

K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

G. Salton and M.H. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill, New York.

B. Schölkopf and A. J. Smola. 2001. *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

C. Strapparava, A. Gliozzo, and C. Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation: Irst at senseval-3. In *Proc. of SENSEVAL-3 Third International Workshop on Evaluation of Systems for the Semantic Analysis of Text*, pages 229–234, Barcelona, Spain, July.

S.K.M. Wong, W. Ziarko, and P.C.N. Wong. 1985. Generalized vector space model in information retrieval. In *Proceedings of the 8^{th} ACM SIGIR Conference*.

S. Zelikovitz and H. Hirsh. 2001. Using LSI for text classification in the presence of background text. In Henrique Paques, Ling Liu, and David Grossman, editors, *Proceedings of CIKM-01, 10th ACM International Conference on Information and Knowledge Management*, pages 113–118, Atlanta, US. ACM Press, New York, US.

Discriminative Training of Clustering Functions: Theory and Experiments with Entity Identification

Xin Li and Dan Roth
Department of Computer Science
University of Illinois, Urbana, IL 61801
(xli1,danr)@cs.uiuc.edu

Abstract

Clustering is an optimization procedure that partitions a set of elements to optimize some criteria, based on a fixed distance metric defined between the elements. Clustering approaches have been widely applied in natural language processing and it has been shown repeatedly that their success depends on defining a good distance metric, one that is appropriate for the task and the clustering algorithm used. This paper develops a framework in which clustering is viewed as a learning task, and proposes a way to train a distance metric that is appropriate for the chosen clustering algorithm in the context of the given task. Experiments in the context of the entity identification problem exhibit significant performance improvements over state-of-the-art clustering approaches developed for this problem.

1 Introduction

Clustering approaches have been widely applied to natural language processing (NLP) problems. Typically, natural language elements (words, phrases, sentences, etc.) are partitioned into non-overlapping classes, based on some distance (or similarity) metric defined between them, in order to provide some level of syntactic or semantic abstraction. A key example is that of class-based language models (Brown et al., 1992; Dagan et al., 1999) where clustering approaches are used in order to partition words, determined to be similar, into sets. This enables estimating more robust statistics since these are computed over collections of "similar" words. A large number of different metrics and algorithms have been experimented with these problems (Dagan et al., 1999; Lee, 1997; Weeds et al., 2004). Similarity between words was also used as a metric in a distributional clustering algorithm in (Pantel and Lin, 2002), and it shows that functionally similar words can be grouped together and even separated to smaller groups based on their senses. At a higher level, (Mann and Yarowsky, 2003) disambiguated personal names by clustering people's home pages using a TFIDF similarity, and several other researchers have applied clustering at the same level in the context of the entity identification problem (Bilenko et al., 2003; McCallum and Wellner, 2003; Li et al., 2004). Similarly, approaches to coreference resolution (Cardie and Wagstaff, 1999) use clustering to identify groups of references to the same entity.

Clustering is an optimization procedure that takes as input (1) a collection of domain elements along with (2) a distance metric between them and (3) an algorithm selected to partition the data elements, with the goal of optimizing some form of clustering quality with respect to the given distance metric. For example, the K-Means clustering approach (Hartigan and Wong, 1979) seeks to maximize a measure of tightness of the resulting clusters based on the Euclidean distance. Clustering is typically called an unsupervised method, since data elements are used without labels during the clustering process and labels are not used to provide feedback to the optimization process. E.g., labels are not taken into account when measuring the quality of the partition. However, in many cases, supervision is used at the application level when determining an appropriate distance metric (e.g., (Lee, 1997; Weeds et al., 2004; Bilenko et al., 2003) and more).

This scenario, however, has several setbacks. First, the process of clustering, simply a function that partitions a set of elements into different classes, involves no learning and thus lacks flexibility. Second, clustering quality is typically defined with respect to a fixed distance metric, without utilizing any direct supervision, so the practical clustering outcome could be disparate from one's intention. Third, when clustering with a given algorithm and a fixed metric, one in fact makes some implicit assumptions on the data and the task (e.g., (Kamvar et al., 2002); more on that below). For example, the optimal conditions under which for K-means works are that the data is generated from a uniform mixture of Gaussian models; this may not hold in reality.

This paper proposes a new clustering framework that addresses all the problems discussed above. Specifically,

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 64–71, Ann Arbor, June 2005.

we define clustering as a learning task: in the training stage, a partition function, parameterized by a distance metric, is trained with respect to a specific clustering algorithm, with supervision. Some of the distinct properties of this framework are that: (1) The training stage is formalized as an optimization problem in which a partition function is learned in a way that minimizes a clustering error. (2) The clustering error is well-defined and driven by feedback from labeled data. (3) Training a distance metric with respect to any given clustering algorithm seeks to minimize the clustering error on training data that, under standard learning theory assumptions, can be shown to imply small error also in the application stage. (4) We develop a general learning algorithm that can be used to learn an expressive distance metric over the feature space (e.g., it can make use of kernels).

While our approach makes explicit use of labeled data, we argue that, in fact, many clustering applications in natural language also exploit this information off-line, when exploring which metrics are appropriate for the task. Our framework makes better use of this resource by incorporating it directly into the metric training process; training is driven by true clustering error, computed via the specific algorithm chosen to partition the data.

We study this new framework empirically on the entity identification problem – identifying whether different mentions of real world entities, such as "JFK" and "John Kennedy", within and across text documents, actually represent the same concept (McCallum and Wellner, 2003; Li et al., 2004). Our experimental results exhibit a significant performance improvement over existing approaches ($20\% - 30\%$ F_1 error reduction) on all three types of entities we study, and indicate its promising prospective in other natural language tasks.

The rest of this paper discusses existing clustering approaches (Sec. 2) and then introduces our Supervised Discriminative Clustering framework (SDC) (Sec. 3) and a general learner for training in it (Sec. 4). Sec. 5 describes the entity identification problem and Sec. 6 compares different clustering approaches on this task.

2 Clustering in Natural Language Tasks

Clustering is the task of partitioning a set of elements $S \subseteq X$ into a disjoint decomposition [1] $p(S) = \{S_1, S_2, \cdots, S_K\}$ of S. We associate with it a *partition function* $p = p_S : X \to C = \{1, 2, \ldots K\}$ that maps each $x \in S$ to a class index $p_S(x) = k$ iff $x \in S_k$. The subscript S in p_S and $p_S(x)$ is omitted when clear from the context. Notice that, unlike a classifier, the image $x \in S$ under a partition function depends on S.

In practice, a clustering algorithm $\mathcal{A}$ (e.g. K-Means), and a distance metric d (e.g., Euclidean distance), are typically used to generate a function h to approximate the true partition function p. Denote $h(S) = \mathcal{A}_d(S)$, the partition of S by h. A distance (equivalently, a similarity) function d that measures the proximity between two elements is a pairwise function $X \times X \to R^+$, which can be parameterized to represent a family of functions — metric properties are not discussed in this paper. For example, given any two element $x_1 =< x_1^{(1)}, \cdots, x_1^{(m)} >$ and $x_2 =< x_2^{(1)}, \cdots, x_2^{(m)} >$ in an m-dimensional space, a linearly weighted Euclidean distance with parameters $\theta = \{w_l\}_1^m$ is defined as:

$$d_\theta(x_1, x_2) \equiv \sqrt{\sum_{l=1}^{m} w_l \cdot |x_1^{(l)} - x_2^{(l)}|^2} \qquad (1)$$

When supervision (e.g. class index of elements) is unavailable, the quality of a partition function h operating on $S \subseteq X$, is measured with respect to the distance metric defined over X. Suppose h partitions S into disjoint sets $h(S) = \{S'_k\}_1^K$, one *quality function* used in the K-Means algorithm is defined as:

$$q_S(h) \equiv \sum_{k=1}^{K} \sum_{x \in S'_k} d(x, \mu'_k)^2, \qquad (2)$$

where μ'_k is the mean of elements in set S'_k. However, this measure can be computed irrespective of the algorithm.

2.1 What is a Good Metric?

A good metric is one in which close proximity correlates well with the likelihood of being in the same class. When applying clustering to some task, people typically decide on the clustering quality measure $q_S(h)$ they want to optimize, and then chose a specific clustering algorithm $\mathcal{A}$ and a distance metric d to generate a 'good' partition function h. However, it is clear that without any supervision, the resulting function is not guaranteed to agree with the target function p (or one's original intention).

Given this realization, there has been some work on *selecting* a good distance metric for a family of related problems and on *learning* a metric for specific tasks. For the former, the focus is on developing and selecting good distance (similarity) metrics that reflect well pairwise proximity between domain elements. The "goodness" of a metric is empirically measured when combined with different clustering algorithms on different problems. For example (Lee, 1997; Weeds et al., 2004) compare similarity metrics such as the Cosine, Manhattan and Euclidean distances, Kullback-Leibler divergence, Jensen-Shannon divergence, and Jaccard's Coefficient, that could be applied in general clustering tasks, on the task of measuring distributional similarity. (Cohen et al., 2003) compares a number of string and token-based similarity metrics on the task of matching entity names and found that,

[1] Overlapping partitions will not be discussed here.

overall, the best-performing method is a hybrid scheme (SoftTFIDF) combining a TFIDF weighting scheme of tokens with the Jaro-Winkler string-distance scheme that is widely used for record linkage in databases.

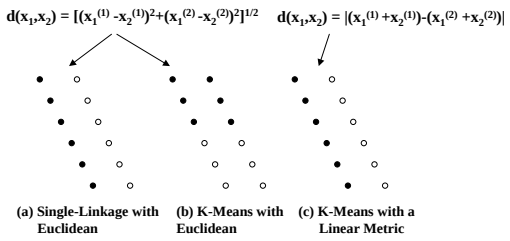


Figure 1: **Different combinations of clustering algorithms with distance metrics.** The 12 points, positioned in a two-dimensional space $< X^{(1)}, X^{(2)} >$, are clustered into two groups containing solid and hollow points respectively.

Moreover, it is not clear whether there exists any *universal* metric that is good for many different problems (or even different data sets for similar problems) and is appropriate for any clustering algorithm. For the word-based distributional similarity mentioned above, this point was discussed in (Geffet and Dagan, 2004) when it is shown that proximity metrics that are appropriate for class-based language models may not be appropriate for other tasks. We illustrate this critical point in Fig. 1. (a) and (b) show that even for the same data collection, different clustering algorithms with the same metric could generate different outcomes. (b) and (c) show that with the same clustering algorithm, different metrics could also produce different outcomes. *Therefore, a good distance metric should be both domain-specific and associated with a specific clustering algorithm.*

2.2 Metric Learning via Pairwise Classification

Several works (Cohen et al., 2003; Cohen and Richman, 2002; McCallum and Wellner, 2003; Li et al., 2004) have tried to remedy the aforementioned problems by attempting to learn a distance function in a domain-specific way via pairwise classification. In the training stage, given a set of labeled element pairs, a function $f : X \times X \rightarrow \{0, 1\}$ is trained to classify any two elements as to whether they belong to the same class (1) or not (0), independently of other elements. The distance between the two elements is defined by converting the prediction confidence of the pairwise classifier, and clustering is then performed based on this distance function. Particularly, (Li et al., 2004) applied this approach to measuring name similarity in the entity identification problem, where a pairwise classifier (LMR) is trained using the SNoW learning architecture (Roth, 1998) based on variations of Perceptron and Winnow, and using a collection of relational features between a pair of names. The distance between two names is defined as a softmax over the classifier's output. As expected, experimental evidence (Cohen et al., 2003; Cohen and Richman, 2002; Li et al., 2004) shows that domain-specific distance functions improve over a fixed metric. This can be explained by the flexibility provided by adapting the metric to the domain as well as the contribution of supervision that guides the adaptation of the metric.

A few works (Xing et al., 2002; Bar-Hillel et al., 2003; Schultz and Joachims, 2004; Mochihashi et al., 2004) outside the NLP domain have also pursued this general direction, and some have tried to learn the metric with limited amount of supervision, no supervision or by incorporating other information sources such as constraints on the class memberships of the data elements. In most of these cases, the algorithm practically used in clustering, (e.g. K-Means), is not considered in the learning procedure, or only implicitly exploited by optimizing the same objective function. (Bach and Jordan, 2003; Bilenko et al., 2004) indeed suggest to learn a metric directly in a clustering task but the learning procedure is specific for one clustering algorithm.

3 Supervised Discriminative Clustering

To solve the limitations of existing approaches, we develop the Supervised Discriminative Clustering Framework (SDC), that can train a distance function with respect to any chosen clustering algorithm in the context of a given task, guided by supervision.

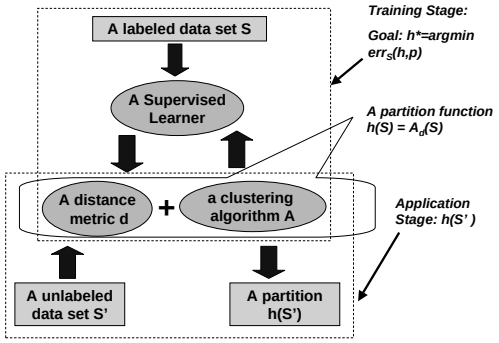


Figure 2: **Supervised Discriminative Clustering**

Fig. 2 presents this framework, in which a clustering task is explicitly split into training and application stages, and the chosen clustering algorithm involves in both stages. In the training stage, supervision is directly integrated into measuring the clustering error $err_S(h, p)$ of a partition function h by exploiting the feedback given by the true partition p. The goal of training is to find a partition function h^* in a hypothesis space H that minimizes the error. Consequently, given a new data set S' in the application stage, under some standard learning theory assumptions, the hope is that the learned partition function

can generalize well and achieve small error as well.

3.1 Supervised and Unsupervised Training

Let p be the target function over X, h be a function in the hypothesis space H, and $h(S) = \{S'_k\}_1^K$. In principle, given data set $S \subseteq X$, if the true partition $p(S) = \{S_k\}_1^K$ of S is available, one can measure the deviation of h from p over S, using an *error function* $err_S(h, p) \rightarrow R^+$. We distinguish an error function from a quality function (as in Equ. 2) as follows: an error function measures the disagreement between clustering and the target partition (or one's intention) when supervision is given, while a quality is defined without any supervision.

For clustering, there is generally no direct way to compare the true class index $p(x)$ of each element with that given by a hypothesis $h(x)$, so an alternative is to measure the disagreement between p and h over pairs of elements. Given a labeled data set S and $p(S)$, one error function, namely *weighted clustering error*, is defined as a sum of the pairwise errors over any two elements in S, weighted by the distance between them:

$$err_S(h,p) \equiv \frac{1}{|S|^2} \sum_{x_i,x_j \in S} [d(x_i,x_j) \cdot A_{ij} + (D - d(x_i,x_j)) \cdot B_{ij}] \quad (3)$$

where $D = \max_{x_i,x_j \in S} d(x_i, x_j)$ is the maximum distance between any two elements in S and I is an indicator function. $A_{ij} \equiv I[(p(x_i) = p(x_j)\ \&\ h(x_i) \neq h(x_j)]$ and $B_{ij} \equiv I[(p(x_i) \neq p(x_j)\ \&\ h(x_i) = h(x_j)]$ represent two types of pairwise errors respectively.

Just like the quality defined in Equ. 2, this error is a function of the metric d. Intuitively, the contribution of a pair of elements that should belong to the same class but are split by h, grows with their distance, and vice versa. However, this measure is significantly different from the quality, in that it does not just measure the tightness of the partition given by h, but rather the difference between the tightness of the partitions given by h and by p.

Given a set of observed data, the goal of training is to learn a good partition function, parameterized by specific clustering algorithms and distance functions. Depending on whether training data is labeled or unlabeled, we can further define supervised and unsupervised training.

Definition 3.1 ***Supervised Training:*** *Given a labeled data set S and $p(S)$, a family of partition functions H, and the error function $err_S(h, p)(h \in H)$, the problem is to find an optimal function h^* s.t.*

$$h^* = argmin_{h \in H}\ err_S(h, p).$$

Definition 3.2 ***Unsupervised Training:*** *Given an unlabeled data set S ($p(S)$ is unknown), a family of partition functions H, and a quality function $q_S(h)(h \in H)$, the problem is to find an optimal partition function h^* s.t.*

$$h^* = argmax_{h \in H}\ q_S(h).$$

With this formalization, SDC along with supervised training, can be distinguished clearly from (1) unsupervised clustering approaches, (2) clustering over pairwise classification; and (3) related works that exploit partial supervision in metric learning as constraints.

3.2 Clustering via Metric Learning

By fixing the clustering algorithm in the training stage, we can further define supervised metric learning, a special case of supervised training.

Definition 3.3 ***Supervised Metric Learning:*** *Given a labeled data set S and $p(S)$, and a family of partition functions $H = \{h\}$ that are parameterized by a chosen clustering algorithm $\mathcal{A}$ and a family of distance metrics d_θ ($\theta \in \Omega$), the problem is to seek an optimal metric d_{θ^*} with respect to $\mathcal{A}$, s.t. for $h(S) = \mathcal{A}_{\ d_\theta}(S)$*

$$\theta^* = argmin_\theta\ err_S(h, p). \quad (4)$$

Learning the metric parameters θ requires parameterizing h as a function of θ, when the algorithm $\mathcal{A}$ is chosen and fixed in h. In the later experiments of Sec. 5, we try to learn weighted Manhattan distances for the single-link algorithm and other algorithms, in the task of entity identification. In this case, when pairwise features are extracted for any elements $x_1, x_2 \in X$, $(x_1, x_2) =< \phi_1, \phi_2, \cdots, \phi_m >$, the *linearly weighted Manhattan distance*, parameterized by ($\theta = \{w_l\}_1^m$) is defined as:

$$d(x_1, x_2) \equiv \sum_{l=1}^{m} w_l \cdot \phi_l(x_1, x_2) \quad (5)$$

where w_l is the weight over feature $\phi_l(x_1, x_2)$. Since measurement of the error is dependent on the metric, as shown in Equ. 3, one needs to enforce some constraints on the parameters. One constraint is $\sum_{l=1}^{m} |w_l| = 1$, which prevents the error from being scale-dependent (e.g., metrics giving smaller distance are always better).

4 A General Learner for SDC

In addition to the theoretical SDC framework, we also develop a practical learning algorithm based on gradient descent (in Fig. 3), that can train a distance function for any chosen clustering algorithm (such as Single-Linkage and K-Means), as in the setting of supervised metric learning. The training procedure incorporates the clustering algorithm (step 2.a) so that the metric is trained with respect to the specific algorithm that will be applied in evaluation. The convergence of this general training procedure depends on the convexity of the error as a function of θ. For example, since the error function we use is *linear* in θ, the algorithm is guaranteed to converge to a global minimum. In this case, for rate of convergence, one can appeal to general results that typically imply, when there exists a parameter vector with zero error, that convergence rate

depends on the 'separation" of the training data, which roughly means the minimal error archived with this parameter vector. Results such as (Freund and Schapire, 1998) can be used to extend the rate of convergence result a bit beyond the separable case, when a small number of the pairs are not separable.

Algorithm: SDC-Learner
Input: S and $p(S)$: the labeled data set. $\mathcal{A}$: the clustering algorithm. $err_S(h, p)$: the clustering error function. $\alpha > 0$: the learning rate. T (typically T is large) : the number of iterations allowed.
Output: θ^* : the parameters in the distance function d.

1. In the initial (I-) step, we randomly choose θ^0 for d. After this step we have the initial d^0 and h^0.
2. Then we iterate over t $(t = 1, 2, \cdots)$,
   (a) Partition S using $h^{t-1}(S) \equiv \mathcal{A}_{d^{t-1}}(S)$;
   (b) Compute $err_S(h^{t-1}, p)$ and update θ using the formula: $\theta^t = \theta^{t-1} - \alpha \cdot \frac{\partial err_S(h^{t-1}, p)}{\partial \theta^{t-1}}$.
   (c) Normalization: $\theta^t = \frac{1}{Z} \cdot \theta^t$, where $Z = ||\theta^t||$.
3. Stopping Criterion: If $t > T$, the algorithm exits and outputs the metric in the iteration with the least error.

Figure 3: **A general training algorithm for SDC**

For the weighted clustering error in Equ. 3, and linearly weighted Manhattan distances as in Equ. 5, the update rule in Step $2(b)$ becomes

$$w_l^t = w_l^{t-1} - \alpha \cdot [\psi_l^{t-1}(p, S) - \psi_l^{t-1}(h, S)]. \quad (6)$$

where $\psi_l(p, S) \equiv \frac{1}{|S|^2} \sum_{x_i, x_j \in S} \phi_l(x_i, x_j) \cdot I[p(x_i) = p(x_j)]$ and $\psi_l(h, S) \equiv \frac{1}{|S|^2} \sum_{x_i, x_j \in S} \phi_l(x_i, x_j) \cdot I[h(x_i) = h(x_j)]$, and $\alpha > 0$ is the learning rate.

5 Entity Identification in Text

We conduct experimental study on the task of entity identification in text (Bilenko et al., 2003; McCallum and Wellner, 2003; Li et al., 2004). A given entity – representing a person, a location or an organization – may be mentioned in text in multiple, ambiguous ways. Consider, for example, an open domain question answering system (Voorhees, 2002) that attempts, given a question like: "When was President Kennedy born?" to search a large collection of articles in order to pinpoint the concise answer: "on May 29, 1917." The sentence, and even the document that contains the answer, may not contain the name "President Kennedy"; it may refer to this entity as "Kennedy", "JFK" or "John Fitzgerald Kennedy". Other documents may state that "John F. Kennedy, Jr. was born on November 25, 1960", but this fact refers to our target entity's son. Other mentions, such as "Senator Kennedy" or "Mrs. Kennedy" are even "closer" to the writing of the target entity, but clearly refer to different entities. Understanding natural language requires identifying whether different mentions of a name, within and across documents, represent the same entity.

We study this problem for three entity types – People, Location and Organization. Although deciding the coreference of names within the same document might be relatively easy, since within a single document identical mentions typically refer to the same entity, identifying coreference across-document is much harder. With no standard corpora for studying the problem in a general setting – both within and across documents, we created our own corpus. This is done by collecting about $8,600$ names from 300 randomly sampled 1998-2000 New York Times articles in the TREC corpus (Voorhees, 2002). These names are first annotated by a named entity tagger, then manually verified and given as input to an entity identifier.

Since the number of classes (entities) for names is very large, standard multi-class classification is not feasible. Instead, we compare SDC with several pairwise classification and clustering approaches. Some of them (for example, those based on SoftTFIDF similarity) do not make use of any domain knowledge, while others do exploit supervision, such as LMR and SDC. Other works (Bilenko et al., 2003) also exploited supervision in this problem by discriminative training of a pairwise classifier but were shown to be inferior.

1. *SoftTFIDF Classifier* – a pairwise classifier deciding whether any two names refer to the same entity, implemented by thresholding a state-of-art SoftTFIDF similarity metric for string comparison (Cohen et al., 2003). Different thresholds have been experimented but only the best results are reported.
2. *LMR Classifier (P|W)* – a SNoW-based pairwise classifier (Li et al., 2004) (described in Sec. 2.2) that learns a linear function for each class over a collection of relational features between two names: including string and token-level features and structural features (listed in Table 1).

   For pairwise classifiers like LMR and SoftTFIDF, prediction is made over pairs of names so transitivity of predictions is not guaranteed as in clustering.
3. *Clustering over SoftTFIDF* – a clustering approach based on the SoftTFIDF similarity metric.
4. *Clustering over LMR (P|W)* – a clustering approach (Li et al., 2004) by converting the LMR classifier into a similarity metric (see Sec. 2.2).
5. *SDC* – our new supervised clustering approach. The distance metric is represented as a linear function over a set of pairwise features as defined in Equ. 5.

The above approaches (2), (4) and (5) learn a classifier or a distance metric using the same feature set as in Table 1. Different clustering algorithms [2], such as Single-Linkage, Complete-Linkage, Graph clustering (George,

[2]The clustering package *Cluster* by Michael Eisen at Stanford University is adopted for K-medoids and *CLUTO* by (George, 2003) is used for other algorithms. Details of these algorithms can be found there.

Honorific Equal	active if both tokens are honorifics and identical.
Honorific Equivalence	active if both tokens are honorifics, not identical, but equivalent.
Honorific Mismatch	active for different honorifics.
Equality	active if both tokens are identical.
Case-Insensitive Equal	active if the tokens are case-insensitive equal.
Nickname	active if tokens have a "nickname" relation.
Prefix Equality	active if the prefixes of both tokens are equal.
Substring	active if one of the tokens is a substring of the other.
Abbreviation	active if one of the tokens is an abbreviation of the other.
Prefix Edit Distance	active if the prefixes of both tokens have an edit-distance of 1.
Edit Distance	active if the tokens have an edit-distance of 1.
Initial	active if one of the tokens is an initial of another.
Symbol Map	active if one token is a symbolic representative of the other.
Structural	recording the location of the tokens that generate other features in two names.

Table 1: **Features employed by LMR and SDC.**

2003) – seeking a minimum cut of a nearest neighbor graph, Repeated Bisections and K-medoids (Chu et al., 2001) (a variation of K-means) are experimented in (5). The number of entities in a data set is always given.

6 Experimental Study

Our experimental study focuses on (1) evaluating the supervised discriminative clustering approach on entity identification; (2) comparing it with existing pairwise classification and clustering approaches widely used in similar tasks; and (3) further analyzing the characteristics of this new framework.

We use the TREC corpus to evaluate different approaches in identifying three types of entities: People, Locations and Organization. For each type, we generate three pairs of training and test sets, each containing about 300 names. We note that the three entity types yield very different data sets, exhibited by some statistical properties[3]. Results on each entity type will be averaged over the three sets and ten runs of two-fold cross-validation for each of them. For SDC, given a training set with annotated name pairs, a distance function is first trained using the algorithm in Fig. 3 (in 20 iterations) with respect to a clustering algorithm and then be used to partition the corresponding test set with the same algorithm.

For a comparative evaluation, the outcomes of each approach on a test set of names are converted to a classification over all possible pairs of names (including non-matching pairs). Only examples in the set M_p, those that are predicated to belong to the same entity (positive predictions) are used in the evaluation, and are compared with the set M_a of examples annotated as positive. The performance of an approach is then evaluated by F_1 value, defined as: $F_1 = \frac{2|M_p \bigcap M_a|}{|M_p|+|M_a|}$.

[3]The average SoftTFIDF similarity between names of the same entity is 0.81, 0.89 and 0.95 for people, locations and organizations respectively.

6.1 Comparison of Different Approaches

Fig. 4 presents the performance of different approaches (described in Sec. 5) on identifying the three entity types. We experimented with different clustering algorithms but only the results by Single-Linkage are reported for *Cluster over LMR (P|W)* and SDC, since they are the best.

SDC works well for all three entity types in spite of their different characteristics. The best F_1 values of SDC are 92.7%, 92.4% and 95.7% for people, locations and organizations respectively, about $20\% - 30\%$ error reduction compared with the best performance of the other approaches. This is an indication that this new approach which integrates metric learning and supervision in a unified framework, has significant advantages [4].

6.2 Further Analysis of SDC

In the next experiments, we will further analyze the characteristics of SDC by evaluating it in different settings.

Different Training Sizes Fig. 5 reports the relationship between the performance of SDC and different training sizes. The learning curves for other learning-based approaches are also shown. We find that SDC exhibits good learning ability with limited supervision. When training examples are very limited, for example, only 10% of all 300 names, pairwise classifiers based on Perceptron and Winnow exhibit advantages over SDC. However, when supervision become reasonable ($30\%+$ examples), SDC starts to outperform all other approaches.

Different Clustering Algorithms Fig. 6 shows the performance of applying different clustering algorithms (see Sec. 5) in the SDC approach. Single-Linkage and Complete-Linkage outperform all other algorithms. One possible reason is that this task has a great number of

[4]We note that in this experiment, the relative comparison between the pairwise classifiers and the clustering approaches over them is not consistent for all entity types. This can be partially explained by the theoretical analysis in (Li et al., 2004) and the difference between entity types.

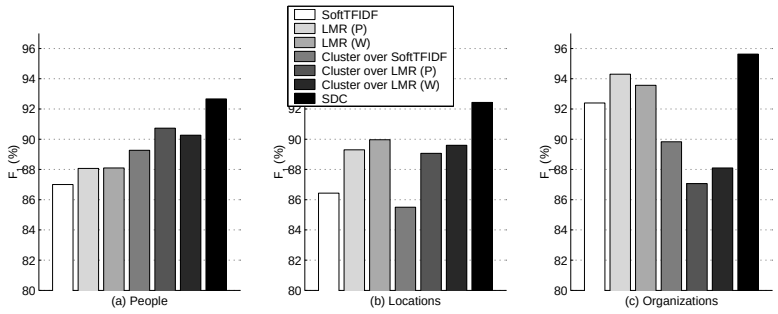


Figure 4: **Performance of different approaches.** The results are reported for SDC with a learning rate $\alpha = 100.0$. The Single-Linkage algorithm is applied whenever clustering is performed. Results are reported in F_1 and averaged over the three data sets for each entity type and 10 runs of two-fold cross-validation. Each training set typically contains 300 annotated names.

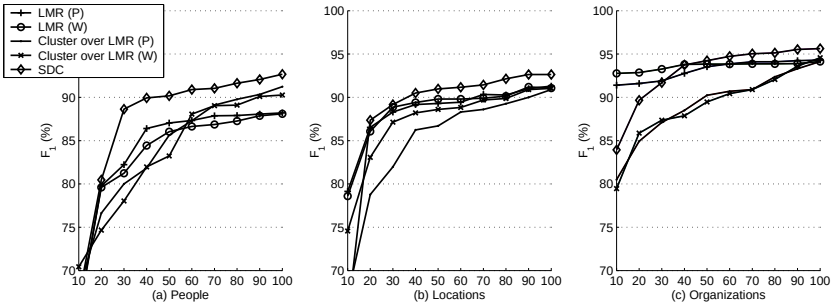


Figure 5: **Performance for different training sizes.** Five learning-based approaches are compared. Single-Linkage is applied whenever clustering is performed. X-axis denotes different percentages of 300 names used in training. Results are reported in F_1 and averaged over the three data sets for each entity type.

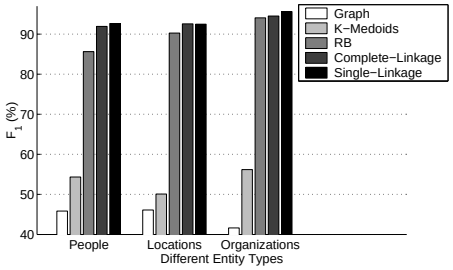


Figure 6: **Different clustering algorithms.** Five clustering algorithms are compared in SDC ($\alpha = 100.0$). Results are averaged over the three data sets for each entity type and 10 runs of two-fold cross-validations.

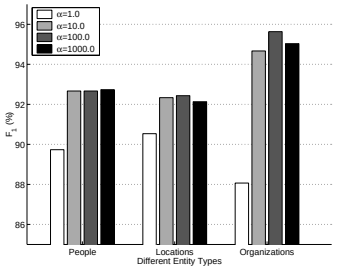


Figure 7: **Performance for different learning rates.** SDC with different learning rates ($\alpha = 1.0, 10.0, 100.0, 1000.0$) compared in this setting. Single-Linkage clustering algorithm is applied.

classes ($100 - 200$ entities) for 300 names in each single data set. The results indicate that the metric learning process relies on properties of the data set, as well as the clustering algorithm. Even if a good distance metric could be learned in SDC, choosing an appropriate algorithm for the specific task is still important.

Different Learning Rates We also experimented with different learning rates in the SDC approach as shown in Fig. 7. It seems that SDC is not very sensitive to different learning rates as long as it is in a reasonable range.

6.3 Discussion

The reason that SDC can outperform existing clustering approaches can be explained by the advantages of SDC – training the distance function with respect to the chosen clustering algorithm, guided by supervision, but they do not explain why it can also outperform the pairwise classifiers. One intuitive explanation is that supervision in the entity identification task or similar tasks is typically given on whether two names correspond to the same entity – entity-level annotation. Therefore it does not necessarily mean whether they are similar in appearance. For exam-

ple, “Brian” and “Wilson” could both refer to a person “Brian Wilson” in different contexts, and thus this name pair is a positive example in training a pairwise classifier. However, with features that only capture the appearance similarity between names, such apparently different names become training noise. This is what exactly happened when we train the LMR classifier with such name pairs. SDC, however, can employ this entity-level annotation and avoid the problem through transitivity in clustering. In the above example, if there is “Brian Wilson” in the data set, then “Brian” and “Wilson” can be both clustered into the same group with “Brian Wilson”. Such cases do not frequently occur for locations and organization but still exist .

7 Conclusion

In this paper, we explicitly formalize clustering as a learning task, and propose a unified framework for training a metric for any chosen clustering algorithm, guided by domain-specific supervision. Our experiments exhibit the advantage of this approach over existing approaches on Entity Identification. Further research in this direction will focus on (1) applying it to more NLP tasks, e.g. coreference resolution; (2) analyzing the related theoretical issues, e.g. the convergence of the algorithm; and (3) comparing it experimentally with related approaches, such as (Xing et al., 2002) and (McCallum and Wellner, 2003).

Acknowledgement This research is supported by NSF grants IIS-9801638 and ITR IIS-0085836, an ONR MURI Award and an equipment donation from AMD.

References

F. R. Bach and M. I. Jordan. 2003. Learning spectral clustering. In *NIPS-03*.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. 2003. Learning distance functions using equivalence relations. In *ICML-03*, pages 11–18.

M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. 2003. Adaptive name matching in information integration. *IEEE Intelligent Systems*, pages 16–23.

M Bilenko, S. Basu, and R. J. Mooney. 2004. Integrating constraints and metric learning in semi-supervised clustering. In *ICML-04*, pages 81–88.

P. Brown, P. deSouza R. Mercer, V. Pietra, and J. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

C. Cardie and K. Wagstaff. 1999. Noun phrase coreference as clustering. In *EMNLP-99*, pages 82–89.

S. C. Chu, J. F. Roddick, and J. S. Pan. 2001. A comparative study and extensions to k-medoids algorithms. In *ICOTA-01*.

W. Cohen and J. Richman. 2002. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD-02*, pages 475–480.

W. Cohen, P. Ravikumar, and S. Fienberg. 2003. A comparison of string metrics for name-matching tasks. In *IIWeb Workshop 2003*, pages 73–78.

I. Dagan, L. Lee, and F. Pereira. 1999. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

Y. Freund and R. Schapire. 1998. Large margin classification using the Perceptron algorithm. In *COLT-98*.

M. Geffet and I. Dagan. 2004. Automatic feature vector quality and distributional similarity. In *COLING-04*.

K. George. 2003. Cluto: A clustering toolkit. Technical report, Dept of Computer Science, University of Minnesota.

J. Hartigan and M. Wong. 1979. A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108.

S. Kamvar, D. Klein, and C. Manning. 2002. Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. In *ICML-02*, pages 283–290.

L. Lee. 1997. *Similarity-Based Approaches to Natural Language Processing*. Ph.D. thesis, Harvard University, Cambridge, MA.

X. Li, P. Morie, and D. Roth. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI-04*, pages 419–424.

G. Mann and D. Yarowsky. 2003. Unsupervised personal name disambiguation. In *CoNLL-03*, pages 33–40.

A. McCallum and B. Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *IJCAI Workshop on Information Integration on the Web*.

D. Mochihashi, G. Kikui, and K. Kita. 2004. Learning nonstructural distance metric by minimum cluster distortions. In *COLING-04*.

P. Pantel and D. Lin. 2002. Discovering word senses from text. In *KDD-02*, pages 613–619.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *AAAI-98*, pages 806–813.

M. Schultz and T. Joachims. 2004. Learning a distance metric from relative comparisons. In *NIPS-04*.

E. Voorhees. 2002. Overview of the TREC-2002 question answering track. In *TREC-02*, pages 115–123.

J. Weeds, D. Weir, and D. McCarthy. 2004. Characterising measures of lexical distributional similarity. In *COLING-04*.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. 2002. Distance metric learning, with application to clustering with side-information. In *NIPS-02*.

Using Uneven Margins SVM and Perceptron for Information Extraction

Yaoyong Li, Kalina Bontcheva and Hamish Cunningham
Department of Computer Science, The University of Sheffield, Sheffield, S1 4DP, UK
{yaoyong,kalina,hamish}@dcs.shef.ac.uk

Abstract

The classification problem derived from information extraction (IE) has an imbalanced training set. This is particularly true when learning from smaller datasets which often have a few positive training examples and many negative ones. This paper takes two popular IE algorithms – SVM and Perceptron – and demonstrates how the introduction of an uneven margins parameter can improve the results on imbalanced training data in IE. Our experiments demonstrate that the uneven margin was indeed helpful, especially when learning from few examples. Essentially, the smaller the training set is, the more beneficial the uneven margin can be. We also compare our systems to other state-of-the-art algorithms on several benchmarking corpora for IE.

1 Introduction

Information Extraction (IE) is the process of automatic extraction of information about pre-specified types of events, entities or relations from text such as newswire articles or Web pages. IE is useful in many applications, such as information gathering in a variety of domains, automatic annotations of web pages for Semantic Web, and knowledge management.

A wide range of machine learning techniques have been used for IE and achieved state-of-the-art results, comparable to manually engineered IE systems. A learning algorithm usually learns a model from a set of documents which have been manually annotated by the user. Then the model can be used to extract information from new documents. Manual annotation is a time-consuming process. Hence, in many cases learning from small data sets is highly desirable. Therefore in this paper we also evaluate the performance of our algorithms on small amounts of training data and show their learning curve.

The learning algorithms for IE can be classified broadly into two main categories: rule learning and statistical learning. The former induces a set of rules from training examples. There are many rule based learning systems, e.g. SRV (Freitag, 1998), RAPIER (Califf, 1998), WHISK (Soderland, 1999), BWI (Freitag and Kushmerick, 2000), and $(LP)^2$ (Ciravegna, 2001). Statistical systems learn a statistical model or classifiers, such as HMMs (Freigtag and McCallum, 1999), Maximal Entropy (Chieu and Ng., 2002), the SVM (Isozaki and Kazawa, 2002; Mayfield et al., 2003), and Perceptron (Carreras et al., 2003). IE systems also differ from each other in the NLP features that they use. These include simple features such as token form and capitalisation information, linguistic features such as part-of-speech, semantic information from gazetteer lists, and genre-specific information such as document structure. In general, the more features the system uses, the better performance it can achieve.

This paper concentrates on classifier-based learning for IE, which typically converts the recognition of each information entity into a set of classification problems. In the framework discussed here, two binary classifiers are trained for each type of information entity. One classifier is used for recognising the entity's start token and the other – the entity's end token.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 72–79, Ann Arbor, June 2005.

The classification problem derived from IE usually has imbalanced training data, in which positive training examples are vastly outnumbered by negative ones. This is particularly true for smaller data sets where often there are hundreds of negative training examples and only few positive ones. Two approaches have been studied so far to deal with imbalanced data in IE. One approach is to under-sample majority class or over-sample minority class in order to obtain a relatively balanced training data (Zhang and Mani, 2003). However, under-sampling can potentially remove certain important examples, and over-sampling can lead to over-fitting and a larger training set. Another approach is to divide the problem into several sub-problems in two layers, each of which has less imbalanced training set than the original one (Carreras et al., 2003; Sitter and Daelemans, 2003). The output of the classifier in the first layer is used as the input to the classifiers in the second layer. As a result, this approach needs more classifiers than the original problem. Moreover, the classification errors in the first layer will affect the performance of the second one.

In this paper we explore another approach to handle the imbalanced data in IE, namely, adapting the learning algorithms for balanced classification to imbalanced data. We particularly study two popular classification algorithms in IE, Support Vector Machines (SVM) and Perceptron.

SVM is a general supervised machine learning algorithm, that has achieved state of the art performance on many classification tasks, including NE recognition. Isozaki and Kazawa (2002) compared three commonly used methods for named entity recognition – the SVM with quadratic kernel, maximal entropy method, and a rule based learning system, and showed that the SVM-based system performed better than the other two. Mayfield et al. (2003) used a lattice-based approach to named entity recognition and employed the SVM with cubic kernel to compute transition probabilities in a lattice. Their results on CoNLL2003 shared task were comparable to other systems but were not the best ones.

Previous research on using SVMs for IE adopts the standard form of the SVM, which treats positive and negative examples equally. As a result, they did not consider the difference between the balanced classification problems, where the SVM performs quite well, and the imbalanced ones. Li and Shawe-Taylor (2003) proposes an uneven margins version of the SVM and shows that the SVM with uneven margins performs significantly better than the standard SVM on document classification problems with imbalanced training data. Since the classification problem for IE is also imbalanced, this paper investigates the SVM with uneven margins for IE tasks and demonstrates empirically that the uneven margins SVM does have better performance than the standard SVM.

Perceptron is a simple, fast and effective learning algorithm, which has successfully been applied to named entity recognition (Carreras et al., 2003). The system uses a two-layer structure of classifiers to handle the imbalanced data. The first layer classifies each word as entity or non-entity. The second layer classifies the named entities identified by the first layer in the respective entity classes. Li et al. (2002) proposed another variant of Perceptron, the Perceptron algorithm with uneven margins (PAUM), designed especially for imbalanced data. In this paper we explore the application of PAUM to IE.

The rest of the paper is structured as follows. Section 2 describes the uneven margins SVM and Perceptron algorithms. Sections 3.1 and 3.2 discuss the classifier-based framework for IE and the experimental datasets we used, respectively. We compare our systems to other state-of-the-art systems on three benchmark datasets in Section 3.3. Section 3.4 discusses the effects of the uneven margins parameter on the SVM and Perceptron's performances. Finally, Section 4 provides some conclusions.

2 Uneven Margins SVM and Perceptron

Li and Shawe-Taylor (2003) introduced an uneven margins parameter into the SVM to deal with imbalanced classification problems. They showed that the SVM with uneven margins outperformed the standard SVM on document classification problem with imbalanced training data. Formally, given a training set $\mathbf{Z} = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$,where $\mathbf{x}_i$ is the n-dimensional input vector and y_i ($= +1$ or -1) its label, the SVM with uneven margins is obtained by solving the quadratic optimisation problem:

$$\min_{\mathbf{w},\, b,\, \xi} \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t.} \quad \langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i + b \geq 1 \quad \text{if } y_i = +1$$

$$\langle \mathbf{w}, \mathbf{x}_i \rangle - \xi_i + b \leq -\tau \quad \text{if } y_i = -1$$

$$\xi_i \geq 0 \qquad \text{for } i = 1, ..., m$$

We can see that the uneven margins parameter τ was added to the constraints of the optimisation problem. τ is the ratio of negative margin to the positive margin of the classifier and is equal to 1 in the standard SVM. For an imbalanced dataset with a few positive examples and many negative ones, it would be beneficial to use larger margin for positive examples than for the negative ones. Li and Shawe-Taylor (2003) also showed that the solution of the above problem could be obtained by solving a related standard SVM problem by, for example, using a publicly available SVM package[1].

Perceptron is an on-line learning algorithm for linear classification. It checks the training examples one by one by predicting their labels. If the prediction is correct, the example is passed; otherwise, the example is used to correct the model. The algorithm stops when the model classifies all training examples correctly. The margin Perceptron not only classifies every training example correctly but also outputs for every training example a value (before thresholding) larger than a predefined parameter (margin). The margin Perceptron has better generalisation capability than the standard Perceptron. Li et al. (2002) proposed the Perceptron algorithm with uneven margins (PAUM) by introducing two margin parameters τ_+ and τ_- into the updating rules for the positive and negative examples, respectively. Similar to the uneven margins parameter in SVM, two margin parameters allow the PAUM to handle imbalanced datasets better than both the standard Perceptron and the margin Perceptron. Additionally, it is known that the Perceptron learning will stop after limited loops only on a linearly separable training set. Hence, a regularisation parameter λ is used in PAUM to guarantee that the algorithm would stop for any training dataset after some updates. PAUM is simple and fast and performed very well on document classification, in particularly on imbalanced training data.

[1]The SVM^{light} package version 3.5, available from http://svmlight.joachims.org/, was used to learn the SVM classifiers in our experiments.

3 Experiments

3.1 Classifier-Based Framework for IE

In the experiments we adopted a classifier-based framework for applying the SVM and PAUM algorithms to IE. The framework consists of three stages: pre-processing of the documents to obtain feature vectors, learning classifiers or applying classifiers to test documents, and finally post-processing the results to tag the documents.

The aim of the preprocessing is to form input vectors from documents. Each document is first processed using the open-source ANNIE system, which is part of GATE[2] (Cunningham et al., 2002). This produces a number of linguistic (NLP) features, including token form, capitalisation information, token kind, lemma, part-of-speech (POS) tag, semantic classes from gazetteers, and named entity types according to ANNIE's rule-based recogniser.

Based on the linguistic information, an input vector is constructed for each token, as we iterate through the tokens in each document (including word, number, punctuation and other symbols) to see if the current token belongs to an information entity or not. Since in IE the context of the token is usually as important as the token itself, the features in the input vector come not only from the current token, but also from preceding and following ones. As the input vector incorporates information from the context surrounding the current token, features from different tokens can be weighted differently, based on their position in the context. The weighting scheme we use is the *reciprocal scheme*, which weights the surrounding tokens reciprocally to the distance to the token in the centre of the context window. This reflects the intuition that the nearer a neighbouring token is, the more important it is for classifying the given token. Our experiments showed that such a weighting scheme obtained better results than the commonly used equal weighting of features (Li et al., 2005).

The key part of the framework is to convert the recognition of information entities into binary classification tasks – one to decide whether a token is the start of an entity and another one for the end token.

After classification, the start and end tags of the

[2]Available from http://www.gate.ac.uk/

entities are obtained and need to be combined into one entity tag. Therefore some post-processing is needed to guarantee tag consistency and to try to improve the results by exploring other information. The currently implemented procedure has three stages. First, in order to guarantee the consistency of the recognition results, the document is scanned from left to right to remove start tags without matching end tags and end tags without preceding start tags. The second stage filters out candidate entities from the output of the first stage, based on their length. Namely, a candidate entity tag is removed if the entity's length (i.e., the number of tokens) is not equal to the length of any entity of the same type in the training set. The third stage puts together all possible tags for a sequence of tokens and chooses the best one according to the probability which was computed from the output of the classifiers (before thresholding) via a Sigmoid function.

3.2 The Experimental Datasets

The paper reports evaluation results on three corpora covering different IE tasks – named entity recognition (CoNLL-2003) and template filling or scenario templates in different domains (Jobs and CFP). The CoNLL-2003[3] provides the most recent evaluation results of many learning algorithms on named entity recognition. The Jobs corpus[4] has also been used recently by several learning systems. The CFP corpus was created as part of the recent Pascal Challenge for evaluation of machine learning methods for IE[5].

In detail, we used the English part of the CoNLL-2003 shared task dataset, which consists of 946 documents for training, 216 document for development (e.g., tuning the parameters in learning algorithm), and 231 documents for evaluation (i.e., testing), all of which are news articles taken from the Reuters English corpus (RCV1). The corpus contains four types of named entities — person, location, organisation and miscellaneous names. In the other two corpora domain-specific information was extracted into a number of slots. The Job corpus includes 300 computer related job advertisements and 17 slots encoding job details, such as title, salary, recruiter, computer language, application, and platform. The CFP corpus consists of 1100 conference or workshop call for papers (CFP), of which 600 were annotated. The corpus includes 11 slots such as workshop and conference names and acronyms, workshop date, location and homepage.

[3] See http://cnts.uia.ac.be/conll2003/ner/

[4] See http://www.isi.edu/info-agents/RISE/repository.html.

[5] See http://nlp.shef.ac.uk/pascal/.

3.3 Comparison to Other Systems

Named Entity Recognition The algorithms are evaluated on the CoNLL-2003 dataset. Since this set comes with development data for tuning the learning algorithm, different settings were tried in order to obtain the best performance on the development set. Different SVM kernel types, window sizes (namely the number of tokens in left or right side of the token at the centre of window), and the uneven margins parameter τ were tested. We found that quadratic kernel, window size 4 and $\tau = 0.5$ produced best results on the development set. These settings were used in all experiments on the CoNLL-2003 dataset in this paper, unless otherwise stated. The parameter settings for PAUM described in Li et al. (2002), e.g. $\tau_+ = 50, \tau_- = 1$, were adopted in all experiments with PAUM, unless otherwise stated.

Table 1 presents the results of our system using three learning algorithms, the uneven margins SVM, the standard SVM and the PAUM on the CONLL-2003 test set, together with the results of three participating systems in the CoNLL-2003 shared task: the best system (Florian et al., 2003), the SVM-based system (Mayfield et al., 2003) and the Perceptron-based system (Carreras et al., 2003).

Firstly, our uneven margins SVM system performed significantly better than the other SVM-based system. As the two systems are different from each other in not only the SVM models used but also other aspects such as the NLP features and the framework, in order to make a fair comparison between the uneven margins SVM and the standard SVM, we also present the results of the two learning algorithms implemented in our framework. We can see from Table 1 that, under the same experimental settings, the uneven margins SVM again performed better than the standard SVM.

Secondly, our PAUM-based system performed slightly better than the system based on voted Perceptron, but there is no significant difference between them. Note that they adopted different mechanisms to deal with the imbalanced data in IE (refer

Table 1: Comparison to other systems on CoNLL-2003 corpus: F-measure(%) on each entity type and the overall micro-averaged F-measure. The 90% confidence intervals for results of other three systems are also presented. The best performance figures for each entity type and overall appear in bold.

	System	LOC	MISC	ORG	PER	Overall
Our Systems	SVM with uneven margins	89.25	77.79	82.29	90.92	86.30
	Standard SVM	88.86	77.32	80.16	88.93	85.05
	PAUM	88.18	76.64	78.26	89.73	84.36
Participating Systems	Best one	**91.15**	**80.44**	**84.67**	**93.85**	**88.76(±0.7)**
	Another SVM	88.77	74.19	79.00	90.67	84.67(±1.0)
	Voted Perceptron	87.88	77.97	80.09	87.31	84.30(±0.9)

to Section 1). The structure of PAUM system is simpler than that of the voted Perceptron system.

Finally, the PAUM system performed worse than the SVM system. On the other hand, training time of PAUM is only 1% of that for the SVM and the PAUM implementation is much simpler than that of SVM. Therefore, when simplicity and speed are required, PAUM presents a good alternative.

Template Filling On *Jobs corpus* our systems are compared to several state-of-the-art learning systems, which include the rule based systems Rapier (Califf, 1998), $(LP)^2$ (Ciravegna, 2001) and BWI (Freitag and Kushmerick, 2000), the statistical system HMM (Freitag and Kushmerick, 2000), and the double classification system (Sitter and Daelemans, 2003). In order to make the comparison as informative as possible, the same settings are adopted in our experiments as those used by $(LP)^2$, which previously reported the highest results on this dataset. In particular, the results are obtained by averaging the performance in ten runs, using a random half of the corpus for training and the rest for testing. Only basic NLP features are used: token form, capitalisation information, token types, and lemmas.

Preliminary experiments established that the SVM with linear kernel obtained better results than SVM with quadratic kernel on the Jobs corpus (Li et al., 2005). Hence we used the SVM with linear kernel in the experiments on the Jobs data. Note that PAUM always uses linear kernel in our experiments.

Table 2 presents the results of our systems as well as the other six systems which have been evaluated on the Jobs corpus. Note that the results for all the 17 slots are available for only three systems, Rapier, $(LP)^2$ and double classification, while the results for some slots were available for the other three systems. We computed the macro-averaged F_1 (the mean of the F_1 of all slots) for our systems as well as for the three fully evaluated systems in order to make a comparison of the overall performance.

Firstly, the overall performance of our two systems is significantly better than the other three fully evaluated systems. The PAUM system achieves the best performance on 5 out of the 17 slots. The SVM system performs best on the other 3 slots. Secondly, the double classification system had much worse overall performance than our systems and other two fully evaluated systems. HMM was evaluated only on two slots. It achieved best result on one slot but was much worse on the other slot than our two systems and some of the others. Finally, somewhat surprisingly, our PAUM system achieves better performance than the SVM system on this dataset. Moreover, the computation time of PAUM is about 1/3 of that of the SVM. Hence, the PAUM system performs quite satisfactory on the Jobs corpus.

Our systems were also evaluated by participating in a Pascal challenge – Evaluating Machine Learning for Information Extraction. The evaluation provided not only the *CFP corpus* but also the linguistic features for all tokens by pre-processing the documents. The main purpose of the challenge was to evaluate machine learning algorithms based on the same linguistic features. The only compulsory task is task1, which used 400 annotated documents for training and other 200 annotated documents for testing. See Ireson and Ciravegna (2005) for a short overview of the challenge. The learning methods explored by the participating systems included LP^2, HMM, CRF, SVM, and a variety of combinations

Table 2: Comparison to other systems on the jobs corpus: F_1 (%) on each entity type and overall performance as macro-averaged F_1. Standard deviations for the MA F_1 of our systems are presented in parenthesis. The highest score on each slot and overall performance appears in bold.

Slot	SVM	PAUM	$(LP)^2$	Rapier	DCs	BWI	HMM	semi-CRF
Id	97.7	97.4	**100**	97.5	97	100	–	–
Title	49.6	53.1	43.9	40.5	35	50.1	**57.7**	40.2
Company	77.2	**78.4**	71.9	70.0	38	78.2	50.4	60.9
Salary	**86.5**	86.4	62.8	67.4	67	–	–	–
Recruiter	78.4	**81.4**	80.6	68.4	55	–	–	–
State	92.8	93.6	84.7	90.2	**94**	–	–	–
City	**95.5**	95.2	93.0	90.4	91	–	–	–
Country	96.2	**96.5**	81.0	93.2	92	–	–	–
Language	86.9	87.3	**91.0**	81.8	33	–	–	–
Platform	80.1	78.4	**80.5**	72.5	36	–	–	–
Application	70.2	69.7	**78.4**	69.3	30	–	–	–
Area	46.8	**54.0**	53.7	42.4	17	–	–	–
Req-years-e	**80.8**	80.0	68.8	67.2	76	–	–	–
Des-years-e	81.9	85.6	60.4	**87.5**	47	–	–	–
Req-degree	87.5	**87.9**	84.7	81.5	45	–	–	–
Des-degree	59.2	62.9	65.1	**72.2**	33	–	–	–
Post date	99.2	99.4	**99.5**	**99.5**	98	–	–	–
MA F_1	80.8($\pm$1.0)	**81.6($\pm$1.1)**	77.2	76.0	57.9	–	–	–

of different learning algorithms. Firstly, the system of the challenge organisers, which is based on LP^2 obtained the best result for Task1, followed by one of our participating systems which combined the uneven margins SVM and PAUM (see Ireson and Ciravegna (2005)). Our SVM and PAUM systems on their own were respectively in the fourth and fifth position among the 20 participating systems. Secondly, at least six other participating system were also based on SVM but used different IE framework and possibly different SVM models from our SVM system. Our SVM system achieved better results than all those SVM-based systems, showing that the SVM models and the IE framework of our system were quite suitable to IE task. Thirdly, our PAUM based system was not as good as our SVM system but was still better than the other SVM based systems. The computation time of the PAUM system was about 1/5 of that of our SVM system.

Table 3 presents the per slot results and overall performance of our SVM and PAUM systems as well as the system with the best overall result. Compared to the best system, our SVM system performed better on two slots and had similar results on many of other slots. The best system had extremely good results on the two slots, C-acronym and C-homepage. Actually, the F_1 values of the best system on the two slots were more than double of those of every other participating system.

3.4 Effects of Uneven Margins Parameter

A number of experiments were conducted to investigate the influence of the uneven margins parameter on the SVM and Perceptron's performances. Table 4 show the results with several different values of uneven margins parameter respectively for the SVM and the Perceptron on two datasets – CoNLL-2003 and Jobs. The SVM with uneven margins ($\tau < 1.0$) had better results than the standard SVM ($\tau = 1$). We can also see that the results were similar for the τ between 0.6 and 0.4, showing that the results are not particularly sensitive to the value of the uneven margins parameter. The uneven margins parameter has similar effect on Perceptron as on the SVM. Table 4 shows that the PAUM had better results than both the standard Perceptron and the margin Perceptron

Table 3: Results of our SVM and PAUM systems on CFP corpus: F-measures(%) on individual entity type and the overall figures, together with the system with the highest overall score. The highest score on each slot appears in bold.

SLOT	PAUM	SVM	Best one
W-name	51.9	**54.2**	35.2
W-acronym	50.4	60.0	**86.5**
W-date	67.0	69.0	**69.4**
W-homepage	69.6	70.5	**72.1**
W-location	60.0	**66.0**	48.8
W-submission	70.2	69.6	**86.4**
W-notification	76.1	85.6	**88.9**
W-camera-ready	71.5	74.7	**87.0**
C-name	43.2	47.7	**55.1**
C-acronym	38.8	38.7	**90.5**
C-homepage	7.1	11.6	**39.3**
Micro-average	61.1	64.3	**73.4**

Our conjecture was that the uneven margins parameter was more helpful on small training sets, because the smaller a training set is, the more imbalanced it could be. Therefore we carried out experiments on a small numbers of training documents. Table 5 shows the results of the SVM and the uneven margins SVM on different numbers of training documents from CoNLL-2003 and Jobs datasets. The performance of both the standard SVM and the uneven margins SVM improves consistently as more training documents are used. Moreover, compared to the results one large training sets shown in Table 4, the uneven margins SVM obtains more improvements on small training sets than the standard SVM model. We can see that the smaller the training set is, the better the results of the uneven margins SVM are in comparison to the standard SVM.

4 Conclusions

This paper studied the uneven margins versions of two learning algorithms – SVM and Perceptron – to deal with the imbalanced training data in IE. Our experiments showed that the uneven margin is helpful, in particular on small training sets. The smaller the training set is, the more beneficial the uneven margin could be. We also showed that the systems based on the uneven margins SVM and Perceptron were comparable to other state-of-the-art systems.

Table 4: The effects of uneven margins parameter of the SVM and Perceptron, respectively: macro averaged $F_1(\%)$ on the two datasets CoNLL-2003 (development set) and Jobs. The standard deviations for the Jobs dataset show the statistical significances of the results. In bold are the best performance figures for each dataset and each system.

τ	1.0	0.8	0.6	0.4	0.2
Conll	89.0	89.6	**89.7**	89.2	85.3
Jobs	79.0	79.9	**81.0**	80.8	79.0
	$\pm$1.4	$\pm$1.2	$\pm$**0.9**	$\pm$1.0	$\pm$1.3
(τ_+, τ_-)	(0,0)	(1,1)	(50,1)		
Conll	83.5	83.9	**84.4**		
Jobs	74.1	78.8	**81.6**		
	$\pm$1.5	$\pm$1.0	$\pm$**1.1**		

Our SVM system obtained better results than other SVM-based systems on the CoNLL-2003 corpus and CFP corpus respectively, while being simpler than most of them. This demonstrates that our SVM system is both effective and efficient.

We also explored PAUM, a simple and fast learning algorithm for IE. The results of PAUM were somehow worse (about 0.02 overall F-measure lower) than those of the SVM on two out of three datasets. On the other hand, PAUM is much faster to train and easier to implement than SVM. It is also worth noting that PAUM outperformed some other learning algorithms. Therefore, even PAUM on its own would be a good learning algorithm for IE. Moreover, PAUM could be used in combination with other classifiers or in the more complicated framework such as the one in Carreras et al. (2003).

Since many other tasks in Natural Language Processing, like IE, often lead to imbalanced classification problems and the SVM has been used widely in Natural Language Learning (NLL), we can expect that the uneven margins SVM and PAUM are likely to obtain good results on other NLL problems as well.

Acknowledgements

This work is supported by the EU-funded SEKT project (http://www.sekt-project.org).

Table 5: The performances of the SVM system with small training sets: macro-averaged $F_1(\%)$ on the two datasets CoNLL-2003 (development set) and Jobs. The uneven margins SVM ($\tau = 0.4$) is compared to the standard SVM model with even margins ($\tau = 1$). The standard deviations are presented for results on the Jobs dataset.

size	10	20	30	40	50
	$\tau = 0.4$				
Conll	60.6	66.4	70.4	72.2	72.8
Jobs	51.6	60.9	65.7	68.6	71.1
	± 2.7	± 2.5	± 2.1	± 1.9	± 2.5
	$\tau = 1$				
Conll	46.2	58.6	65.2	68.3	68.6
Jobs	47.1	56.5	61.4	65.4	68.1
	± 3.4	± 3.1	± 2.7	± 1.9	± 2.1

References

M. E. Califf. 1998. *Relational Learning Techniques for Natural Language Information Extraction*. Ph.D. thesis, University of Texas at Austin.

X. Carreras, L. Màrquez, and L. Padró. 2003. Learning a perceptron-based named entity chunker via online recognition feedback. In *Proceedings of CoNLL-2003*, pages 156–159. Edmonton, Canada.

H. L. Chieu and H. T. Ng. 2002. A Maximum Entropy Approach to Information Extraction from Semi-Structured and Free Text. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 786–791.

F. Ciravegna. 2001. $(\text{LP})^2$, an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named Entity Recognition through Classifier Combination. In *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada.

D. Freigtag and A. K. McCallum. 1999. Information Extraction with HMMs and Shrinkage. In *Proceesings of Workshop on Machine Learnig for Information Extraction*, pages 31–36.

D. Freitag and N. Kushmerick. 2000. Boosted Wrapper Induction. In *Proceedings of AAAI 2000*.

D. Freitag. 1998. *Machine Learning for Information Extraction in Informal Domains*. Ph.D. thesis, Carnegie Mellon University.

N. Ireson and F. Ciravegna. 2005. Pascal Challenge The Evaluation of Machine Learning for Information Extraction. In *Proceedings of Dagstuhl Seminar Machine Learning for the Semantic Web (http://www.smi.ucd.ie/Dagstuhl-MLSW/proceedings/)*.

H. Isozaki and H. Kazawa. 2002. Efficient Support Vector Classifiers for Named Entity Recognition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 390–396, Taipei, Taiwan.

Y. Li and J. Shawe-Taylor. 2003. The SVM with Uneven Margins and Chinese Document Categorization. In *Proceedings of The 17th Pacific Asia Conference on Language, Information and Computation (PACLIC17)*, Singapore, Oct.

Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola. 2002. The Perceptron Algorithm with Uneven Margins. In *Proceedings of the 9th International Conference on Machine Learning (ICML-2002)*, pages 379–386.

Y. Li, K. Bontcheva, and H. Cunningham. 2005. SVM Based Learning System For Information Extraction. In *Proceedings of Sheffield Machine Learning Workshop*, Lecture Notes in Computer Science. Springer Verlag.

J. Mayfield, P. McNamee, and C. Piatko. 2003. Named Entity Recognition Using Hundreds of Thousands of Features. In *Proceedings of CoNLL-2003*, pages 184–187. Edmonton, Canada.

A. De Sitter and W. Daelemans. 2003. Information extraction via double classification. In *Proceedings of ECML/PRDD 2003 Workshop on Adaptive Text Extraction and Mining (ATEM 2003)*, Cavtat-Dubrovnik, Croatia.

S. Soderland. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272.

J. Zhang and I. Mani. 2003. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*.

Improving sequence segmentation learning by predicting trigrams

Antal van den Bosch
ILK / Computational Linguistics and AI
Tilburg University
Tilburg, The Netherlands
Antal.vdnBosch@uvt.nl

Walter Daelemans
CNTS, Department of Linguistics
University of Antwerp
Antwerp, Belgium
Walter.Daelemans@ua.ac.be

Abstract

Symbolic machine-learning classifiers are known to suffer from near-sightedness when performing sequence segmentation (chunking) tasks in natural language processing: without special architectural additions they are oblivious of the decisions they made earlier when making new ones. We introduce a new pointwise-prediction single-classifier method that predicts trigrams of class labels on the basis of windowed input sequences, and uses a simple voting mechanism to decide on the labels in the final output sequence. We apply the method to maximum-entropy, sparse-winnow, and memory-based classifiers using three different sentence-level chunking tasks, and show that the method is able to boost generalization performance in most experiments, attaining error reductions of up to 51%. We compare and combine the method with two known alternative methods to combat near-sightedness, viz. a feedback-loop method and a stacking method, using the memory-based classifier. The combination with a feedback loop suffers from the label bias problem, while the combination with a stacking method produces the best overall results.

1 Optimizing output sequences

Many tasks in natural language processing have the full sentence as their domain. Chunking tasks, for example, deal with segmenting the full sentence into chunks of some type, for example constituents or named entities, and possibly labeling each identified chunk. The latter typically involves disambiguation among alternative labels (e.g. syntactic role labeling, or semantic type assignment). Both tasks, whether seen as separate tasks or as one, involve the use of contextual knowledge from the available input (e.g. words with part-of-speech tags), but also the coordination of segmentations and disambiguations over the sentence as a whole.

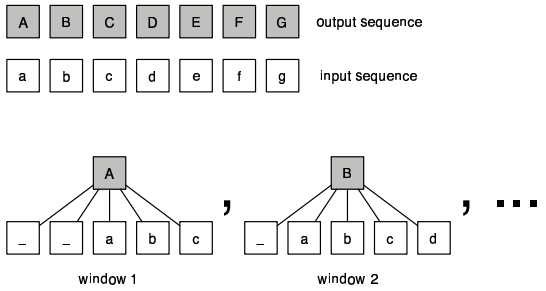


Figure 1: Standard windowing process. Sequences of input symbols and output symbols are converted into windows of fixed-width input symbols each associated with one output symbol.

Many machine-learning approaches to chunking tasks use windowing, a standard representational approach to generate cases that can be sequentially processed. Each case produces one element of the output sequence. The simplest method to process these cases is that each case is classified in isolation, generating a so-called point-wise prediction; the sequence of subsequent predictions can be concatenated to form the entire output analysis of the sentence. Within a window, fixed-width subsequences of adjacent input symbols, representing a certain contextual scope, are mapped to one output symbol, typically associated with one of the input symbols, for example the middle one. Figure 1 displays this standard version of the windowing process.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 80–87, Ann Arbor, June 2005.

The fact that the point-wise classifier is only trained to associate subsequences of input symbols to single output symbols as accurately as possible is a problematic restriction: it may easily cause the classifier to produce invalid or impossible output sequences, since it is incapable of taking into account any decisions it has made earlier. This well-known problem has triggered at least the following three main types of solutions.

Feedback loop Each training or test example may represent not only the regular windowed input, but also a copy of previously made classifications, to allow the classifier to be more consistent with its previous decisions. Direct feedback loops that copy a predicted output label to the input representation of the next example have been used in symbolic machine-learning architectures such as the the maximum-entropy tagger described by Ratnaparkhi (1996) and the memory-based tagger (MBT) proposed by Daelemans et al. (1996). This solution assumes that processing is directed, e.g. from left to right. A noted problem of this approach is the *label bias problem* (Lafferty et al., 2001), which is that a feedback-loop classifier may be driven to be consistent with its previous decision also in the case this decision was wrong; sequences of errors may result.

Stacking, boosting, and voting The partly incorrect concatenated output sequence of a single classifier may serve as input to a second-stage classifier in a stacking architecture, a common machine-learning optimization technique (Wolpert, 1992). Although less elegant than a monolithic single-classifier architecture, this method is known to be capable of recognizing recurring errors of the first-stage classifier and correcting them (Veenstra, 1998). Boosting (Freund and Schapire, 1996) has been applied to optimize chunking systems (Carreras et al., 2002), as well as voting over sets of different classifiers (Florian et al., 2003). Punyakanok and Roth (2001) present two methods for combining the predictions of different classifiers according to constraints that ensure that the resulting output is made more coherent.

Output sequence optimization Rather than basing classifications only on model parameters estimated from co-occurrences between input and output symbols employed for maximizing the likelihood of point-wise single-label predictions at the output level, classifier output may be augmented by an optimization over the output sequence as a whole using optimization techniques such as beam searching in the space of a conditional markov model's output (Ratnaparkhi, 1996) or hidden markov models (Skut and Brants, 1998). Maximum-entropy markov models (McCallum et al., 2000) and conditional random fields (Lafferty et al., 2001) optimize the likelihood of segmentations of output symbol sequences through variations of Viterbi search. A non-stochastic, non-generative method for output sequence optimization is presented by Argamon et al. (1999), who propose a memory-based sequence learner that finds alternative chunking analyses of a sequence, and produces one best-guess analysis by a tiling algorithm that finds an optimal joining of the alternative analyses.

In this paper we introduce a symbolic machine-learning method that can be likened to the approaches of the latter type of output sequence optimizers, but which does not perform a search in a space of possible analyses. The approach is to have a point-wise symbolic machine-learning classifier predict series of overlapping n-grams (in the current study, trigrams) of class symbols, and have a simple voting mechanism decide on the final output sequence based on the overlapping predicted trigrams. We show that the approach has similar positive effects when applied to a memory-based classifier and a maximum-entropy classifier, while yielding mixed effects with a sparse-winnow classifier. We then proceed to compare the trigram prediction method to a feedback-loop method and a stacking method applied using the memory-based classifier. The three methods attain comparable error reductions. Finally, we combine the trigram-prediction method with each of the two other methods. We show that the combination of the trigram-prediction method and the feedback-loop method does not improve performance due to the label bias problem. In contrast, the combination of the trigram-prediction method and the stacking method leads to the overall best results, indicating that the latter two methods solve complementary aspects of the near-sightedness problem.

The structure of the paper is as follows. First,

we introduce the three chunking sequence segmentation tasks studied in this paper and explain the automatic algorithmic model selection method for the three machine-learning classifiers used in our study, in Section 2. The subsequent three sections report on empirical results for the different methods proposed for correcting the near-sightedness of classifiers: the new class-trigrams method, a feedback-loop approach in combination with single classes and class trigrams, and two types of stacking in combination with single classes and class trigrams. Section 6 sums up and discusses the main results of the comparison.

2 Data and methodology

The three data sets we used for this study represent a varied set of sentence-level chunking tasks of both syntactic and semantic nature: English base phrase chunking (henceforth CHUNK), English named-entity recognition (NER), and disfluency chunking in transcribed spoken Dutch utterances (DISFL).

CHUNK is the task of splitting sentences into non-overlapping syntactic phrases or constituents. The used data set, extracted from the WSJ Penn Treebank, contains 211,727 training examples and 47,377 test instances. The examples represent seven-word windows of words and their respective (predicted) part-of-speech tags, and each example is labeled with a class using the IOB type of segmentation coding as introduced by Ramshaw and Marcus (1995), marking whether the middle word is inside (I), outside (O), or at the beginning (B) of a chunk. Words occuring less than ten times in the training material are attenuated (converted into a more general string that retains some of the word's surface form). Generalization performance is measured by the F-score on correctly identified and labeled constituents in test data, using the evaluation method originally used in the "shared task" subevent of the CoNLL-2000 conference (Tjong Kim Sang and Buchholz, 2000) in which this particular training and test set were used. An example sentence with base phrases marked and labeled is the following: [*He*]$_{NP}$ [*reckons*]$_{VP}$ [*the current account deficit*]$_{NP}$ [*will narrow*]$_{VP}$ [*to*]$_{PP}$ [*only $ 1.8 billion*]$_{NP}$ [*in*]$_{PP}$ [*September*]$_{NP}$.

NER, named-entity recognition, is to recognize and type named entities in text. We employ the English NER shared task data set used in the CoNLL-2003 conference, again using the same evaluation method as originally used in the shared task (Tjong Kim Sang and De Meulder, 2003). This data set discriminates four name types: persons, organizations, locations, and a rest category of "miscellany names". The data set is a collection of newswire articles from the Reuters Corpus, RCV1[1]. The given training set contains 203,621 examples; as test set we use the "testb" evaluation set which contains 46,435 examples. Examples represent seven-word windows of unattenuated words with their respective predicted part-of-speech tags. No other task-specific features such as capitalization identifiers or seed list features were used. Class labels use the IOB segmentation coding coupled with the four possible name type labels. Analogous to the CHUNK task, generalization performance is measured by the F-score on correctly identified and labeled named entities in test data. An example sentence with the named entities segmented and typed is the following: [*U.N.*]$_{organization}$ *official* [*Ekeus*]$_{person}$ *heads for* [*Baghdad*]$_{location}$.

DISFL, disfluency chunking, is the task of recognizing subsequences of words in spoken utterances such as fragmented words, laughter, self-corrections, stammering, repetitions, abandoned constituents, hesitations, and filled pauses, that are not part of the syntactic core of the spoken utterance. We use data introduced by Lendvai et al. (2003), who extracted the data from a part of the Spoken Dutch Corpus of spontaneous speech[2] that is both transcribed and syntactically annotated. All words and multi-word subsequences judged not to be part of the syntactic tree are defined as disfluent chunks. We used a single 90% – 10% split of the data, producing a training set of 303,385 examples and a test set of 37,160 examples. Each example represents a window of nine words (attenuated below an occurrence threshold of 100) and 22 binary features representing various string overlaps (to encode possible repetitions); for details, cf. (Lendvai

[1] Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19.

[2] CGN, Spoken Dutch Corpus, version 1.0, `http://lands.let.kun.nl/cgn/ehome.htm`.

et al., 2003). Generalization performance is measured by the F-score on correctly identified disfluent chunks in test data. An example of a chunked Spoken Dutch Corpus sentence is the following ("uh" is a filled pause; without the disfluencies, the sentence means "I have followed this process with a certain amount of scepticism for about a year"): [*ik uh*] *ik heb met de nodige scepsis* [*uh*] *deze gang van zaken* [*zo'n*] *zo'n jaar aangekeken*.

We perform our experiments on the three tasks using three machine-learning algorithms: the memory-based learning or k-nearest neighbor algorithm as implemented in the TiMBL software package (version 5.1) (Daelemans et al., 2004), henceforth referred to as MBL; maximum-entropy classification (Guiasu and Shenitzer, 1985) as implemented in the maxent software package (version 20040930) by Zhang Le[3], henceforth MAXENT; and a sparse-winnow network (Littlestone, 1988) as implemented in the SNoW software package (version 3.0.5) by Carlson et al. (1999), henceforth WINNOW. All three algorithms have algorithmic parameters that bias their performance; to allow for a fair comparison we optimized each algorithm on each task using wrapped progressive sampling (Van den Bosch, 2004) (WPS), a heuristic automatic procedure that, on the basis of validation experiments internal to the training material, searches among algorithmic parameter combinations for a combination likely to yield optimal generalization performance on unseen data. We used wrapped progressive sampling in all experiments.

3 Predicting class trigrams

There is no intrinsic bound to what is packed into a class label associated to a windowed example. For example, complex class labels can span over trigrams of singular class labels. A classifier that learns to produce trigrams of class labels will at least produce syntactically valid trigrams from the training material, which might partly solve some near-sightedness problems of the single-class classifier. Although simple and appealing, the lurking disadvantage of the trigram idea is that the number of class labels increases explosively when moving from single class labels to wider trigrams. The CHUNK data, for example, has 22 classes ("IOB" codes associated with chunk types); in the same training set, 846 different trigrams of these 22 classes and the start/end context symbol occur. The eight original classes of NER combine to 138 occurring trigrams. DISFL only has two classes, but 18 trigram classes.

[3]Maximum Entropy Modeling Toolkit for Python and C++, `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`.

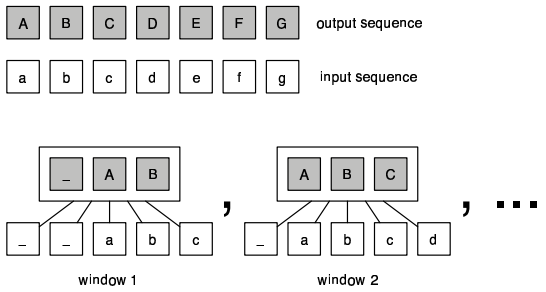


Figure 2: Windowing process with trigrams of class symbols. Sequences of input symbols and output symbols are converted into windows of fixed-width input symbols each associated with, in this example, trigrams of output symbols.

Figure 2 illustrates the procedure by which windows are created with, as an example, class trigrams. Each windowed instance maps to a class label that incorporates three atomic class labels, namely the focus class label that was the original unigram label, plus its immediate left and right neighboring class labels.

While creating instances this way is trivial, it is not entirely trivial how the output of overlapping class trigrams recombines into a normal string of class sequences. When the example illustrated in Figure 2 is followed, each single class label in the output sequence is effectively predicted three times; first, as the right label of a trigram, next as the middle label, and finally as the left label. Although it would be possible to avoid overlaps and classify only every three words, there is an interesting property of overlapping class label n-grams: it is possible to vote over them. To pursue our example of trigram classes, the following voting procedure can be followed to decide about the resulting unigram class label sequence:

1. When all three votes are unanimous, their common class label is returned;
2. When two out of three votes are for the same

	MBL			MAXENT			WINNOW		
Task	Baseline	Trigram	red.	Baseline	Trigram	red.	Baseline	Trigram	red.
CHUNK	91.9	**92.7**	10	90.3	**91.9**	17	**89.5**	88.3	-11
NER	77.2	**80.2**	17	47.5	**74.5**	51	68.9	**70.1**	4
DISFL	77.9	**81.7**	17	75.3	**80.7**	22	**70.5**	65.3	-17

Table 1: Comparison of generalization performances of three machine-learning algorithms in terms of F-score on the three test sets without and with class trigrams. Each third column displays the error reduction in F-score by the class trigrams method over the other method. The best performances per task are printed in bold.

class label, this class label is returned;

3. When all three votes disagree (i.e., when majority voting ties), the class label is returned of which the classifier is most confident.

Classifier confidence, needed for the third tie-breaking rule, can be heuristically estimated by taking the distance of the nearest neighbor in MBL, the estimated probability value of the most likely class produced by the MAXENT classifier, or the activation level of the most active unit of the WINNOW network.

Clearly this scheme is one out of many possible schemes, using variants of voting as well as variants of n (and having multiple classifiers with different n, so that some back-off procedure could be followed). For now we use this procedure with trigrams as an example. To measure its effect we apply it to the sequence tasks CHUNK, NER, and DISFL. The results of this experiment, where in each case WPS was used to find optimal algorithmic parameters of all three algorithms, are listed in Table 1. We find rather positive effects of the trigram method both with MBL and MAXENT; we observe relative error reductions in the F-score on chunking ranging between 10% and a remarkable 51% error reduction, with MAXENT on the NER task. With WINNOW, we observe decreases in performance on CHUNK and DISFL, and a minor error reduction of 4% on NER.

4 The feedback-loop method versus class trigrams

An alternative method for providing a classifier access to its previous decisions is a feedback-loop approach, which extends the windowing approach by feeding previous decisions of the classifier as features into the current input of the classifier. This approach was proposed in the context of memory-based learning for part-of-speech tagging as MBT (Daelemans et al., 1996). The number of decisions fed back into the input can be varied. In the experiments described here, the feedback loop iteratively updates a memory of the three most recent predictions.

Task	Baseline	Feedback	Trigrams	Feed+Tri
CHUNK	91.9	**93.0**	92.7	89.8
NER	77.2	78.1	**80.2**	77.5
DISFL	77.9	78.6	**81.7**	79.1

Table 2: Comparison of generalization performances in terms of F-score of MBL on the three test sets, with and without a feedback loop, and the error reduction attained by the feedback-loop method, the F-score of the trigram-class method, and the F-score of the combination of the two methods.

The feedback-loop approach can be combined both with single class and class trigram output. In the latter case, the full trigram class labels are copied to the input, retaining at any time the three most recently predicted labels in the input. Table 2 shows the results for both options on the three chunking tasks. The feedback-loop method outperforms the trigram-class method on CHUNK, but not on the other two tasks. It does consistently outperform the baseline single-class classifier. Interestingly, the combination of the two methods performs worse than the baseline classifier on CHUNK, and also performs worse than the trigram-class method on the other two tasks.

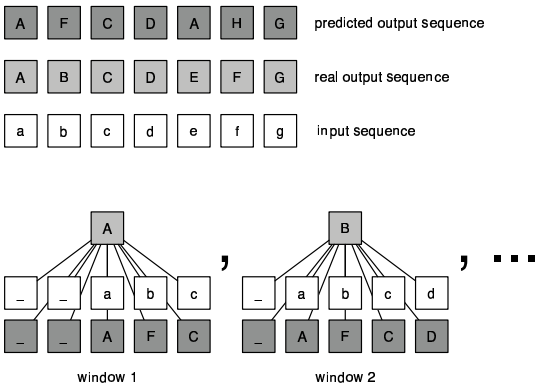


Figure 3: The windowing process after a first-stage classifier has produced a predicted output sequence. Sequences of input symbols, predicted output symbols, and real output symbols are converted into windows of fixed-width input symbols and predicted output symbols, each associated with one output symbol.

5 Stacking versus class trigrams

Stacking, a term popularized by Wolpert (1992) in an artificial neural network context, refers to a class of meta-learning systems that learn to correct errors made by lower-level classifiers. We implement stacking by adding a windowed sequence of previous and subsequent output class labels to the original input features (here, we copy a window of seven predictions to the input, centered around the middle position), and providing these enriched examples as training material to a second-stage classifier. Figure 3 illustrates the procedure. Given the (possibly erroneous) output of a first classifier on an input sequence, a certain window of class symbols from that predicted sequence is copied to the input, to act as predictive features for the real class label.

To generate the output of a first-stage classifier, two options are open. We name these options **perfect** and **adaptive**. They differ in the way they create training material for the second-stage classifier:

Perfect – the training material is created straight from the training material of the first-stage classifier, by windowing over the real class sequences. In doing so, the class label of each window is excluded from the input window, since it is always the same as the class to be predicted. In training, this focus feature would receive an unrealistically high weight, especially considering that in testing this feature would contain errors. To assign a very high weight to a feature that may contain an erroneous value does not seem a good idea in view of the label bias problem.

Task	Baseline	Perfect stacking	Adaptive stacking
CHUNK	91.9	92.0	**92.6**
NER	77.2	78.3	**78.9**
DISFL	77.9	80.5	**81.6**

Table 3: Comparison of generalization performances in terms of F-score of MBL on the three test sets, without stacking, and with perfect and adaptive stacking.

Adaptive – the training material is created indirectly by running an internal 10-fold cross-validation experiment on the first-stage training set, concatenating the predicted output class labels on all of the ten test partitions, and converting this output to class windows. In contrast with the perfect variant, we do include the focus class feature in the copied class label window. The adaptive approach can in principle learn from recurring classification errors in the input, and predict the correct class in case an error re-occurs.

Table 3 lists the comparative results on the CHUNK, NER, and DISFL tasks introduced earlier. They show that both types of stacking improve performance on the three tasks, and that the adaptive stacking variant produces higher relative gains than the perfect variant; in terms of error reduction in F-score as compared to the baseline single-class classifier, the gains are 9% for CHUNK, 7% for NER, and 17% for DISFL. There appears to be more useful information in training data derived from cross-validated output with errors, than in training data with error-free material.

Stacking and class trigrams can be combined. One possible straightforward combination is that of a first-stage classifier that predicts trigrams, and a second-stage stacked classifier that also predicts trigrams (we use the adaptive variant, since it produced the best results), while including a centered seven-positions-wide window of first-stage trigram class labels in the input. Table 4 compares the results

Task	Adaptive stacking	Trigram	Combination
CHUNK	92.6	92.8	**93.1**
NER	78.9	80.2	**80.6**
DISFL	81.6	81.7	**81.9**

Table 4: Comparison of generalization performances in terms of F-score by MBL on the three test sets, with adaptive stacking, trigram classes, and the combination of the two.

of adaptive stacking and trigram classes with those of the combination of the two. As can be seen, the combination produces even better results than both the stacking and the trigram-class methods individually, on all three tasks. Compared to the baseline single-class classifier, the error reductions are 15% for CHUNK, 15% for NER, and 18% for DISFL.

As an additional analysis, we inspected the predictions made by the trigram-class method and its combinations with the stacking and the feedback-loop methods on the CHUNK task to obtain a better view on the amount of disagreements between the trigrams. We found that with the trigram-class method, in 6.3% of all votes some disagreement among the overlapping trigrams occurs. A slightly higher percentage of disagreements, 7.1%, is observed with the combination of the trigram-class and the stacking method. Interestingly, in the combination of the trigram-class and feedback-loop methods, only 0.1% of all trigram votes are not unanimous. This clearly illustrates that in the latter combination the resulting sequence of trigrams is internally very consistent – also in its errors.

6 Conclusion

Classifiers trained on chunking tasks that make isolated. near-sighted decisions on output symbols and that do not optimize the resulting output sequences afterwards or internally through a feedback loop, tend to produce weak models for sequence processing tasks. To combat this weakness, we have proposed a new method that uses a single symbolic machine-learning classifier predicting trigrams of classes, using a simple voting mechanism to reduce the sequence of predicted overlapping trigrams to a sequence of single output symbols. Compared to their near-sighted counterparts, error reductions are attained of 10 to 51% with MBL and MAXENT on three chunking tasks. We found weaker results with a WINNOW classifier, suggesting that the latter is more sensitive to the division of the class space in more classes, likely due to the relatively sparser co-occurrences between feature values and class labels on which WINNOW network connection weights are based.

We have contrasted the trigram-class method against a feedback-loop method (MBT) and a stacking method, all using a memory-based classifier (but the methods generalize to any machine-learning classifier). With the feedback-loop method, modest error reductions of 3%, 4%, and 17% are measured; stacking attains comparable improvements of 7%, 9%, and 17% error reductions in the chunking F-score. We then combined the trigram-class method with the two other methods. The combination with the feedback-loop system led to relatively low performance results. A closer analysis indicated that the two methods appear to render each other ineffective: by feeding back predicted trigrams in the input, the classifier is very much geared towards predicting a next trigram that will be in accordance with the two partly overlapping trigrams in the input, as suggested by overwhelming evidence in this direction in training material – this problem is also known as the label bias problem (Lafferty et al., 2001). (The fact that maximum-entropy markov models also suffer from this problem prompted Lafferty *et al.* to propose conditional random fields.)

We also observed that the positive effects of the trigram-class and stacking variants do not mute each other when combined. The overall highest error reductions are attained with the combination: 15% for CHUNK, 15% for NER, and 18% for DISFL. The combination of the two methods solve more errors than the individual methods do. Apparently, they both introduce complementary disagreements in overlapping trigrams, which the simple voting mechanism can convert to more correct predictions than the two methods do individually.

Further research should focus on a deep quantitative and qualitative analysis of the different errors the different methods correct when compared to the baseline single-class classifier, as well as the errors they may introduce. Alternatives to the

IOB-style encoding should also be incorporated in these experiments (Tjong Kim Sang, 2000). Additionally, a broader comparison with point-wise predictors (Kashima and Tsuboi, 2004) as well as Viterbi-based probabilistic models (McCallum et al., 2000; Lafferty et al., 2001; Sha and Pereira, 2003) in large-scale comparative studies is warranted. Also, the scope of the study may be broadened to all sequential language processing tasks, including tasks in which no segmentation takes place (e.g. part-of-speech tagging), and tasks at the morpho-phonological level (e.g. grapheme-phoneme conversion and morphological analysis).

Acknowledgements

The authors wish to thank Sander Canisius for discussions and suggestions. The work of the first author is funded by NWO, the Netherlands Organisation for Scientific Research; the second author's work is partially funded by the EU BioMinT project.

References

S. Argamon, I. Dagan, and Y. Krymolowski. 1999. A memory-based approach to learning shallow natural language patterns. *Journal of Experimental and Theoretical Artificial Intelligence*, 10:1–22.

A. J. Carlson, C. M. Cumby, J. L. Rosen, and D. Roth. 1999. Snow user guide. Technical Report UIUCDCS-R-99-2101, Cognitive Computation Group, Computer Science Department, University of Illinois, Urbana, Illinois.

X. Carreras, L. Màrques, and L. Padró. 2002. Named entity extraction using AdaBoost. In *Proceedings of CoNLL-2002*, pages 167–170. Taipei, Taiwan.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: A memory-based part of speech tagger generator. In E. Ejerhed and I. Dagan, editors, *Proceedings of WVLC*, pages 14–27. ACL SIGDAT.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2004. TiMBL: Tilburg memory based learner, version 5.1.0, reference guide. Technical Report ILK 04-02, ILK Research Group, Tilburg University.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named entity recognition through classifier combination. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada.

Y. Freund and R. E. Schapire. 1996. Experiments with a new boosting algorithm. In L. Saitta, editor, *Proceedings of ICML-96*, pages 148–156, San Francisco, CA. Morgan Kaufmann.

S. Guiasu and A. Shenitzer. 1985. The principle of maximum entropy. *The Mathematical Intelligencer*, 7(1).

H. Kashima and Y. Tsuboi. 2004. Kernel-based discriminative learning algorithms for labeling sequences, trees and graphs. In *Proceedings of ICML-2004*, Banff, Canada.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, Williamstown, MA.

P. Lendvai, A. van den Bosch, and E. Krahmer. 2003. Memory-based disfluency chunking. In *Proceedings of DISS'03)*, Gothenburg, Sweden, pages 63–66.

N. Littlestone. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of ICML-00*, Stanford, CA.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. The MIT Press.

L.A. Ramshaw and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of WVLC-95*, Cambridge, MA, pages 82–94.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP, May 17-18, 1996, University of Pennsylvania*.

F. Sha and F. Pereira. 2003. Shallow parsing with Conditional Random Fields. In *Proceedings of HLT-NAACL 2003*, Edmonton, Canada.

W. Skut and T. Brants. 1998. Chunk tagger: statistical recognition of noun phrases. In *ESSLLI-1998 Workshop on Automated Acquisition of Syntax and Parsing*.

E. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132.

E. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

E. Tjong Kim Sang. 2000. Noun phrase recognition by system combination. In *Proceedings of ANLP-NAACL 2000*, pages 50–55. Seattle, Washington, USA. Morgan Kaufman Publishers.

A. van den Bosch. 2004. Wrapped progressive sampling search for optimizing learning algorithm parameters. In R. Verbrugge, N. Taatgen, and L. Schomaker, editors, *Proceedings of the 16th Belgian-Dutch AI Conference*, pages 219–226, Groningen, The Netherlands.

J. Veenstra. 1998. Fast NP chunking using memory-based learning techniques. In *Proceedings of BENELEARN'98*, pages 71–78, Wageningen, The Netherlands.

D. H. Wolpert. 1992. Stacked Generalization. *Neural Networks*, 5:241–259.

An Expectation Maximization Approach to Pronoun Resolution

Colin Cherry and **Shane Bergsma**
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{colinc,bergsma}@cs.ualberta.ca

Abstract

We propose an unsupervised Expectation Maximization approach to pronoun resolution. The system learns from a fixed list of potential antecedents for each pronoun. We show that unsupervised learning is possible in this context, as the performance of our system is comparable to supervised methods. Our results indicate that a probabilistic gender/number model, determined automatically from unlabeled text, is a powerful feature for this task.

1 Introduction

Coreference resolution is the process of determining which expressions in text refer to the same real-world entity. Pronoun resolution is the important yet challenging subset of coreference resolution where a system attempts to establish coreference between a pronominal anaphor, such as a third-person pronoun like *he, she, it,* or *they*, and a preceding noun phrase, called an antecedent. In the following example, a pronoun resolution system must determine the correct antecedent for the pronouns "his" and "he."

(1) When the president entered the arena with his family, he was serenaded by a mariachi band.

Pronoun resolution has applications across many areas of Natural Language Processing, particularly in the field of information extraction. Resolving a pronoun to a noun phrase can provide a new interpretation of a given sentence, giving a Question Answering system, for example, more data to consider.

Our approach is a synthesis of linguistic and statistical methods. For each pronoun, a list of antecedent candidates derived from the parsed corpus is presented to the Expectation Maximization (EM) learner. Special cases, such as pleonastic, reflexive and cataphoric pronouns are dealt with linguistically during list construction. This allows us to train on and resolve all third-person pronouns in a large Question Answering corpus. We learn lexicalized gender/number, language, and antecedent probability models. These models, tied to individual words, can not be learned with sufficient coverage from labeled data. Pronouns are resolved by choosing the most likely antecedent in the candidate list according to these distributions. The resulting resolution accuracy is comparable to supervised methods.

We gain further performance improvement by initializing EM with a gender/number model derived from special cases in the training data. This model is shown to perform reliably on its own. We also demonstrate how the models learned through our unsupervised method can be used as features in a supervised pronoun resolution system.

2 Related Work

Pronoun resolution typically employs some combination of constraints and preferences to select the antecedent from preceding noun phrase candidates. Constraints filter the candidate list of improbable antecedents, while preferences encourage selection of antecedents that are more recent, frequent, etc. Implementation of constraints and preferences can be based on empirical insight (Lappin and Leass, 1994), or machine learning from a reference-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 88–95, Ann Arbor, June 2005.

annotated corpus (Ge et al., 1998). The majority of pronoun resolution approaches have thus far relied on manual intervention in the resolution process, such as using a manually-parsed corpus, or manually removing difficult non-anaphoric cases; we follow Mitkov et al.'s approach (2002) with a fully-automatic pronoun resolution method. Parsing, noun-phrase identification, and non-anaphoric pronoun removal are all done automatically.

Machine-learned, fully-automatic systems are more common in noun phrase coreference resolution, where the method of choice has been decision trees (Soon et al., 2001; Ng and Cardie, 2002). These systems generally handle pronouns as a subset of all noun phrases, but with limited features compared to systems devoted solely to pronouns. Kehler used Maximum Entropy to assign a probability distribution over possible noun phrase coreference relationships (1997). Like his approach, our system does not make hard coreference decisions, but returns a distribution over candidates.

The above learning approaches require annotated training data for supervised learning. Cardie and Wagstaff developed an unsupervised approach that partitions noun phrases into coreferent groups through clustering (1999). However, the partitions they generate for a particular document are not useful for processing new documents, while our approach learns distributions that can be used on unseen data. There are also approaches to anaphora resolution using unsupervised methods to extract useful information, such as gender and number (Ge et al., 1998), or contextual role-knowledge (Bean and Riloff, 2004). Co-training can also leverage unlabeled data through weakly-supervised reference resolution learning (Müller et al., 2002). As an alternative to co-training, Ng and Cardie (2003) use EM to augment a supervised coreference system with unlabeled data. Their feature set is quite different, as it is designed to generalize from the data in a labeled set, while our system models individual words. We suspect that the two approaches can be combined.

Our approach is inspired by the use of EM in bilingual word alignment, which finds word-to-word correspondences between a sentence and its translation. The prominent statistical methods in this field are unsupervised. Our methods are most influenced by IBM's Model 1 (Brown et al., 1993).

3 Methods

3.1 Problem formulation

We will consider our training set to consist of (p, k, C) triples: one for each pronoun, where p is the pronoun to be resolved, k is the pronoun's context, and C is a candidate list containing the nouns p could potentially be resolved to. Initially, we take k to be the parsed sentence that p appears in.

C consists of all nouns and pronouns that precede p, looking back through the current sentence and the sentence immediately preceding it. This small window may seem limiting, but we found that a correct candidate appeared in 97% of such lists in a labeled development text. Mitkov et al. also limit candidate consideration to the same window (2002). Each triple is processed with non-anaphoric pronoun handlers (Section 3.3) and linguistic filters (Section 3.4), which produce the final candidate lists.

Before we pass the (p, k, C) triples to EM, we modify them to better suit our EM formulation. There are four possibilities for the gender and number of third-person pronouns in English: masculine, feminine, neutral and plural (e.g., *he, she, it, they*). We assume a noun is equally likely to corefer with any member of a given gender/number category, and reduce each p to a category label accordingly. For example, *he*, *his*, *him* and *himself* are all labeled as *masc* for masculine pronoun. Plural, feminine and neutral pronouns are handled similarly. We reduce the context term k to p's immediate syntactic context, including only p's syntactic parent, the parent's part of speech, and p's relationship to the parent, as determined by a dependency parser. Incorporating context only through the governing constituent was also done in (Ge et al., 1998). Finally, each candidate in C is augmented with ordering information, so we know how many nouns to "step over" before arriving at a given candidate. We will refer to this ordering information as a candidate's j term, for jump. Our example sentence in Section 1 would create the two triples shown in Figure 1, assuming the sentence began the document it was found in.

3.2 Probability model

Expectation Maximization (Dempster et al., 1977) is a process for filling in unobserved data probabilistically. To use EM to do unsupervised pronoun reso-

his:	p = *masc* k = p's family C = arena (0), president (1)
he:	p = *masc* k = serenade p C = family (0), *masc* (1), arena (2), president (3)

Figure 1: EM input for our example sentence. j-values follow each lexical candidate.

lution, we phrase the resolution task in terms of hidden variables of an observed process. We assume that in each case, one candidate from the candidate list is selected as the antecedent before p and k are generated. EM's role is to induce a probability distribution over candidates to maximize the likelihood of the (p, k) pairs observed in our training set:

$$\Pr(Dataset) = \prod_{(p,k) \in Dataset} \Pr(p, k) \quad (1)$$

We can rewrite $\Pr(p, k)$ so that it uses a hidden candidate (or antecedent) variable c that influences the observed p and k:

$$\Pr(p, k) = \sum_{c \in C} \Pr(p, k, c) \quad (2)$$

$$\Pr(p, k, c) = \Pr(p, k|c)\Pr(c) \quad (3)$$

To improve our ability to generalize to future cases, we use a naïve Bayes assumption to state that the choices of pronoun and context are conditionally independent, given an antecedent. That is, once we select the word the pronoun represents, the pronoun and its context are no longer coupled:

$$\Pr(p, k|c) = \Pr(p|c)\Pr(k|c) \quad (4)$$

We can split each candidate c into its lexical component l and its jump value j. That is, $c = (l, j)$. If we assume that l and j are independent, and that p and k each depend only on the l component of c, we can combine Equations 3 and 4 to get our final formulation for the joint probability distribution:

$$\Pr(p, k, c) = \Pr(p|l)\Pr(k|l)\Pr(l)\Pr(j) \quad (5)$$

The jump term j, though important when resolving pronouns, is not likely to be correlated with any lexical choices in the training set.

Table 1: Examples of learned pronoun probabilities.

Word (l)	*masc*	*fem*	*neut*	*plur*
company	0.03	0.01	0.95	0.01
president	0.94	0.01	0.03	0.02
teacher	0.19	0.71	0.09	0.01

This results in four models that work together to determine the likelihood of a given candidate. The $\Pr(p|l)$ distribution measures the likelihood of a pronoun given an antecedent. Since we have collapsed the observed pronouns into groups, this models a word's affinity for each of the four relevant gender/number categories. We will refer to this as our **pronoun model**. $\Pr(k|l)$ measures the probability of the syntactic relationship between a pronoun and its parent, given a prospective antecedent for the pronoun. This is effectively a **language model**, grading lexical choice by context. $\Pr(l)$ measures the probability that the word l will be found to be an antecedent. This is useful, as some entities, such as "president" in newspaper text, are inherently more likely to be referenced with a pronoun. Finally, $\Pr(j)$ measures the likelihood of jumping a given number of noun phrases backward to find the correct candidate. We represent these models with table look-up. Table 1 shows selected l-value entries in the $\Pr(p|l)$ table from our best performing EM model. Note that the probabilities reflect biases inherent in our news domain training set.

Given models for the four distributions above, we can assign a probability to each candidate in C according to the observations p and k; that is, $\Pr(c|p, k)$ can be obtained by dividing Equation 5 by Equation 2. Remember that $c = (l, j)$.

$$\Pr(c|p, k) = \frac{\Pr(p|l)\Pr(k|l)\Pr(l)\Pr(j)}{\sum_{c' \in C} \Pr(p|l')\Pr(k|l')\Pr(l')\Pr(j')} \quad (6)$$

$\Pr(c|p, k)$ allows us to get fractional counts of (p, k, c) triples in our training set, as if we had actually observed c co-occurring with (p, k) in the proportions specified by Equation 6. This estimation process is effectively the E-step in EM.

The M-step is conducted by redefining our models according to these fractional counts. For example, after assigning fractional counts to candidates

according to $\Pr(c|p,k)$, we re-estimate $\Pr(p|l)$ with the following equation for a specific (p,l) pair:

$$\Pr(p|l) = \frac{N(p,l)}{N(l)} \quad (7)$$

where $N()$ counts the number of times we see a given event or joint event throughout the training set.

Given trained models, we resolve pronouns by finding the candidate $\hat{c}$ that is most likely for the current pronoun, that is $\hat{c} = \text{argmax}_{c \in C} \Pr(c|p,k)$. Because $\Pr(p,k)$ is constant with respect to c, $\hat{c} = \text{argmax}_{c \in C} \Pr(p,k,c)$.

3.3 Non-anaphoric Pronouns

Not every pronoun in text refers anaphorically to a preceding noun phrase. There are a frequent number of difficult cases that require special attention, including pronouns that are:

- Pleonastic: pronouns that have a grammatical function but do not reference an entity. E.g. "It is important to observe it is raining."
- Cataphora: pronouns that reference a future noun phrase. E.g. "In his speech, the president praised the workers."
- Non-noun referential: pronouns that refer to a verb phrase, sentence, or implicit concept. E.g. "John told Mary they should buy a car."

If we construct them naïvely, the candidate lists for these pronouns will be invalid, introducing noise in our training set. Manual handling or removal of these cases is infeasible in an unsupervised approach, where the input is thousands of documents. Instead, pleonastics are identified syntactically using an extension of the detector developed by Lappin and Leass (1994). Roughly 7% of all pronouns in our labeled test data are pleonastic. We detect cataphora using a pattern-based method on parsed sentences, described in (Bergsma, 2005b). Future nouns are only included when cataphora are identified. This approach is quite different from Lappin and Leass (1994), who always include all future nouns from the current sentence as candidates, with a constant penalty added to possible cataphoric resolutions. The cataphora module identifies 1.4% of test data pronouns to be cataphoric; in each instance this identification is correct. Finally, we know of no approach that handles pronouns referring to verb phrases or implicit entities. The unavoidable errors for these pronouns, occurring roughly 4% of the time, are included in our final results.

3.4 Candidate list modifications

It would be possible for C to include every noun phrase in the current and previous sentence, but performance can be improved by automatically removing improbable antecedents. We use a standard set of constraints to filter candidates. If a candidate's gender or number is known, and does not match the pronoun's, the candidate is excluded. Candidates with known gender include other pronouns, and names with gendered designators (such as "Mr." or "Mrs."). Our parser also identifies plurals and some gendered first names. We remove from C all times, dates, addresses, monetary amounts, units of measurement, and pronouns identified as pleonastic.

We use the syntactic constraints from Binding Theory to eliminate candidates (Haegeman, 1994). For the reflexives *himself, herself, itself* and *themselves*, this allows immediate syntactic identification of the antecedent. These cases become unambiguous; only the indicated antecedent is included in C.

We improve the quality of our training set by removing known noisy cases before passing the set to EM. For example, we anticipate that sentences with quotation marks will be problematic, as other researchers have observed that quoted text requires special handling for pronoun resolution (Kennedy and Boguraev, 1996). Thus we remove pronouns occurring in the same sentences as quotes from the learning process. Also, we exclude triples where the constraints removed all possible antecedents, or where the pronoun was deemed to be pleonastic. Performing these exclusions is justified for training, but in testing we state results for all pronouns.

3.5 EM initialization

Early in the development of this system, we were impressed with the quality of the pronoun model $\Pr(p|l)$ learned by EM. However, we found we could construct an even more precise pronoun model for common words by examining unambiguous cases in our training data. Unambiguous cases are pronouns having only one word in their candidate list C. This could be a result of the preprocessors described in

Sections 3.3 and 3.4, or the pronoun's position in the document. A $\Pr_U(p|l)$ model constructed from only unambiguous examples covers far fewer words than a learned model, but it rarely makes poor gender/number choices. Furthermore, it can be obtained without EM. Training on unambiguous cases is similar in spirit to (Hindle and Rooth, 1993). We found in our development and test sets that, after applying filters, roughly 9% of pronouns occur with unambiguous antecedents.

When optimizing a probability function that is not concave, the EM algorithm is only guaranteed to find a local maximum; therefore, it can be helpful to start the process near the desired end-point in parameter space. The unambiguous pronoun model described above can provide such a starting point. When using this **initializer**, we perform our initial E-step by weighting candidates according to $\Pr_U(p|l)$, instead of weighting them uniformly. This biases the initial E-step probabilities so that a strong indication of the gender/number of a candidate from unambiguous cases will either boost the candidate's chances or remove it from competition, depending on whether or not the predicted category matches that of the pronoun being resolved.

To deal with the sparseness of the $\Pr_U(p|l)$ distribution, we use add-1 smoothing (Jeffreys, 1961). The resulting effect is that words with few unambiguous occurrences receive a near-uniform gender/number distribution, while those observed frequently will closely match the observed distribution. During development, we also tried clever initializers for the other three models, including an extensive language model initializer, but none were able to improve over $\Pr_U(p|l)$ alone.

3.6 Supervised extension

Even though we have justified Equation 5 with reasonable independence assumptions, our four models may not be combined optimally for our pronoun resolution task, as the models are only approximations of the true distributions they are intended to represent. Following the approach in (Och and Ney, 2002), we can view the right-hand-side of Equation 5 as a special case of:

$$\exp\left(\begin{array}{c}\lambda_1 \log \Pr(p|l) + \lambda_2 \log \Pr(k|l) + \\ \lambda_3 \log \Pr(l) + \lambda_4 \log \Pr(j)\end{array}\right) \quad (8)$$

where $\forall i : \lambda_i = 1$. Effectively, the log probabilities of our models become feature functions in a log-linear model. When labeled training data is available, we can use the Maximum Entropy principle (Berger et al., 1996) to optimize the λ weights.

This provides us with an optional supervised extension to the unsupervised system. Given a small set of data that has the correct candidates indicated, such as the set we used while developing our unsupervised system, we can re-weight the final models provided by EM to maximize the probability of observing the indicated candidates. To this end, we follow the approach of (Och and Ney, 2002) very closely, including their handling of multiple correct answers. We use the limited memory variable metric method as implemented in Malouf's maximum entropy package (2002) to set our weights.

4 Experimental Design

4.1 Data sets

We used two training sets in our experiments, both drawn from the AQUAINT Question Answering corpus (Vorhees, 2002). For each training set, we manually labeled pronoun antecedents in a corresponding **key** containing a subset of the pronouns in the set. These keys are drawn from a collection of complete documents. For each document, all pronouns are included. With the exception of the supervised extension, the keys are used only to validate the resolution decisions made by a trained system. Further details are available in (Bergsma, 2005b).

The development set consists of 333,000 pronouns drawn from 31,000 documents. The development key consists of 644 labeled pronouns drawn from 58 documents; 417 are drawn from sentences without quotation marks. The development set and its key were used to guide us while designing the probability model, and to fine-tune EM and smoothing parameters. We also use the development key as labeled training data for our supervised extension.

The test set consists of 890,000 pronouns drawn from 50,000 documents. The test key consists of 1209 labeled pronouns drawn from 118 documents; 892 are drawn from sentences without quotation marks. All of the results reported in Section 5 are determined using the test key.

4.2 Implementation Details

To get the context values and implement the syntactic filters, we parsed our corpora with Minipar (Lin, 1994). Experiments on the development set indicated that EM generally began to overfit after 2 iterations, so we stop EM after the second iteration, using the models from the second M-step for testing. During testing, ties in likelihood are broken by taking the candidate closest to the pronoun.

The EM-produced models need to be smoothed, as there will be unseen words and unobserved (p, l) or (k, l) pairs in the test set. This is because problematic cases are omitted from the training set, while all pronouns are included in the key. We handle out-of-vocabulary events by replacing words or context-values that occur only once during training with a special **unknown** symbol. Out-of-vocabulary events encountered during testing are also treated as unknown. We handle unseen pairs with additive smoothing. Instead of adding 1 as in Section 3.5, we add $\delta_p = 0.00001$ for (k, l) pairs, and $\delta_w = 0.001$ for (p, l) pairs. These δ values were determined experimentally with the development key.

4.3 Evaluation scheme

We evaluate our work in the context of a fully automatic system, as was done in (Mitkov et al., 2002). Our evaluation criteria is similar to their *resolution etiquette*. We define accuracy as the proportion of pronouns correctly resolved, either to any coreferent noun phrase in the candidate list, or to the pleonastic category, which precludes resolution. Systems that handle and state performance for all pronouns in unrestricted text report much lower accuracy than most approaches in the literature. Furthermore, automatically parsing and pre-processing texts causes consistent degradation in performance, regardless of the accuracy of the pronoun resolution algorithm. To have a point of comparison to other fully-automatic approaches, note the resolution etiquette score reported in (Mitkov et al., 2002) is 0.582.

5 Results

5.1 Validation of unsupervised method

The key concern of our work is whether enough useful information is present in the pronoun's category, context, and candidate list for unsupervised learning of antecedents to occur. To that end, our first set of experiments compare the pronoun resolution accuracy of our EM-based solutions to that of a previous-noun baseline on our test key. The results are shown in Table 2. The columns split the results into three cases: all pronouns with no exceptions; all cases where the pronoun was found in a sentence containing no quotation marks (and therefore resembling the training data provided to EM); and finally all pronouns excluded by the second case. We compare the following methods:

1. **Previous noun**: Pick the candidate from the filtered list with the lowest j value.
2. **EM, no initializer**: The EM algorithm trained on the test set, starting from a uniform E-step.
3. **Initializer, no EM**: A model that ranks candidates using only a pronoun model built from unambiguous cases (Section 3.5).
4. **EM w/ initializer**: As in (2), but using the initializer in (3) for the first E-step.
5. **Maxent extension**: The models produced by (4) are used as features in a log-linear model trained on the development key (Section 3.6).
6. **Upper bound**: The percentage of cases with a correct answer in the filtered candidate list.

For a reference point, picking the previous noun before applying any of our candidate filters receives an accuracy score of 0.281 on the "All" task.

Looking at the "All" column in Table 2, we see EM can indeed learn in this situation. Starting from uniform parameters it climbs from a 40% baseline to a 60% accurate model. However, the initializer can do slightly better with precise but sparse gender/number information alone. As we hoped, combining the initializer and EM results in a statistically significant[1] improvement over EM with a uniform starting point, but it is not significantly better than the initializer alone. The advantage of the EM process is that it produces multiple models, which can be re-weighted with maximum entropy to reach our highest accuracy, roughly 67%. The λ weights that achieve this score are shown in Table 3.

Maximum entropy leaves the pronoun model $\Pr(p|l)$ nearly untouched and drastically reduces the

[1] Significance is determined throughout Section 5 using McNemar's test with a significance level $\alpha = 0.05$.

Table 2: Accuracy for all cases, all excluding sentences with quotes, and only sentences with quotes.

Method	All	No“ ”	Only“ ”
1 Previous noun	0.397	0.399	0.391
2 EM, no initializer	0.610	0.632	0.549
3 Initializer, no EM	0.628	0.642	0.587
4 EM w/ initializer	0.632	0.663	0.546
5 Maxent extension	0.669	0.696	0.593
6 Upper bound	0.838	0.868	0.754

Table 3: Weights set by maximum entropy.

Model	$\Pr(p\|l)$	$\Pr(k\|l)$	$\Pr(l)$	$\Pr(j)$
Lambda	0.931	0.056	0.070	0.167

Table 4: Comparison to SVM.

Method	Accuracy
Previous noun	0.398
EM w/ initializer	0.664
Maxent extension	0.708
SVM	0.714

influence of all other models (Table 3). This, combined with the success of the initializer alone, leads us to believe that a strong notion of gender/number is very important in this task. Therefore, we implemented EM with several models that used only pronoun category, but none were able to surpass the initializer in accuracy on the test key. One factor that might help explain the initializer's success is that despite using only a $\Pr_U(p|l)$ model, the initializer also has an implicit factor resembling a $\Pr(l)$ model: when two candidates agree with the category of the pronoun, add-1 smoothing ensures the more frequent candidate receives a higher probability.

As was stated in Section 3.4, sentences with quotations were excluded from the learning process because the presence of a correct antecedent in the candidate list was less frequent in these cases. This is validated by the low upper bound of 0.754 in the only-quote portion of the test key. We can see that all methods except for the previous noun heuristic score noticeably better when ignoring those sentences that contain quotation marks. In particular, the difference between our three unsupervised solutions ((2), (3) and (4)) are more pronounced. Much of the performance improvements that correspond to our model refinements are masked in the overall task because adding the initializer to EM does not improve EM's performance on quotes at all. Developing a method to construct more robust candidate lists for quotations could improve our performance on these cases, and greatly increase the percentage of pronouns we are training on for a given corpus.

5.2 Comparison to supervised system

We put our results in context by comparing our methods to a recent supervised system. The comparison system is an SVM that uses 52 linguistically-motivated features, including probabilistic gender/number information obtained through web queries (Bergsma, 2005a). The SVM is trained with 1398 separate labeled pronouns, the same training set used in (Bergsma, 2005a). This data is also drawn from the news domain. Note the supervised system was not constructed to handle all pronoun cases, so non-anaphoric pronouns were removed from the test key and from the candidate lists in the test key to ensure a fair comparison. As expected, this removal of difficult cases increases the performance of our system on the test key (Table 4). Also note there is no significant difference in performance between our supervised extension and the SVM. The completely unsupervised EM system performs worse, but with only a 7% relative reduction in performace compared to the SVM; the previous noun heuristic shows a 44% reduction.

5.3 Analysis of upper bound

If one accounts for the upper bound in Table 2, our methods do very well on those cases where a correct answer actually appears in the candidate list: the best EM solution scores 0.754, and the supervised extension scores 0.800. A variety of factors result in the 196 candidate lists that do not contain a true antecedent. 21% of these errors arise from our limited candidate window (Section 3.1). Incorrect pleonastic detection accounts for another 31% while non-

noun referential pronouns cause 25% (Section 3.3). Linguistic filters (Section 3.4) account for most of the remainder. An improvement in any of these components would result in not only higher final scores, but cleaner EM training data.

6 Conclusion

We have demonstrated that unsupervised learning is possible for pronoun resolution. We achieve accuracy of 63% on an all-pronoun task, or 75% when a true antecedent is available to EM. There is now motivation to develop cleaner candidate lists and stronger probability models, with the hope of surpassing supervised techniques. For example, incorporating antecedent context, either at the sentence or document level, may boost performance. Furthermore, the lexicalized models learned in our system, especially the pronoun model, are potentially powerful features for any supervised pronoun resolution system.

References

David L. Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *HLT-NAACL*, pages 297–304.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Shane Bergsma. 2005a. Automatic acquisition of gender information for anaphora resolution. In *Proceedings of the 18th Conference of the Canadian Society for Computational Intelligence (Canadian AI 2005)*, pages 342–353, Victoria, BC.

Shane Bergsma. 2005b. Corpus-based learning for pronominal anaphora resolution. Master's thesis, Department of Computing Science, University of Alberta, Edmonton. http://www.cs.ualberta.ca/~bergsma/Pubs/thesis.pdf.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

Claire Cardie and Kiri Wagstaff. 1999. Noun phrase coreference as clustering. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–171.

L. Haegeman. 1994. *Introduction to Government & Binding theory: Second Edition*. Basil Blackwell, Cambridge, UK.

Donald Hindle and Mats Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

Harold Jeffreys, 1961. *Theory of Probability*, chapter 3.23. Oxford: Clarendon Press, 3rd edition.

Andrew Kehler. 1997. Probabilistic coreference in information extraction. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 163–173.

Christopher Kennedy and Branimir Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th Conference on Computational Linguistics*, pages 113–118.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Dekang Lin. 1994. Principar - an efficient, broad-coverage, principle-based parser. In *Proceedings of COLING-94*, pages 42–48, Kyoto, Japan.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pages 49–55.

Ruslan Mitkov, Richard Evans, and Constantin Orasan. 2002. A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 168–186.

Christoph Müller, Stefan Rapp, and Michael Strube. 2002. Applying co-training to reference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 352–359.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Vincent Ng and Claire Cardie. 2003. Weakly supervised natural language learning without redundant views. In *HLT-NAACL 2003: Proceedings of the Main Conference*, pages 173–180.

Franz J. Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Philadelphia, PA, July.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Ellen Vorhees. 2002. Overview of the TREC 2002 question answering track. In *Proceedings of the Eleventh Text REtrieval Conference (TREC)*.

Probabilistic Head-Driven Parsing for Discourse Structure

Jason Baldridge and Alex Lascarides
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh, EH8 9LW
Scotland, UK
{jbaldrid,alex}@inf.ed.ac.uk

Abstract

We describe a data-driven approach to building interpretable discourse structures for appointment scheduling dialogues. We represent discourse structures as headed trees and model them with probabilistic head-driven parsing techniques. We show that dialogue-based features regarding turn-taking and domain specific goals have a large positive impact on performance. Our best model achieves an f-score of 43.2% for labelled discourse relations and 67.9% for unlabelled ones, significantly beating a right-branching baseline that uses the most frequent relations.

1 Introduction

Achieving a model of discourse interpretation that is both robust and deep is a major challenge. Consider the dialogue in Figure 1 (the sentence numbers are from the Redwoods treebank (Oepen et al., 2002)). A robust and deep interpretation of it should resolve the anaphoric temporal description in utterance 154 to the twenty sixth of *July* in the afternoon. It should identify that time and before 3pm on the twenty-seventh as potential times to meet, while ruling out July thirtieth to August third. It should gracefully handle incomplete or ungrammatical utterances like 152 and recognise that utterances 151 and 152 have no overall effect on the time and place to meet.

According to Hobbs et al. (1993) and Asher and Lascarides (2003), a discourse structure consisting of hierarchical rhetorical connections between utterances is vital for providing a *unified model* of a wide range of anaphoric and intentional discourse phenomena, contributing to the interpretations of pronouns, temporal expressions, presuppositions and ellipses (among others), as well as influencing communicative goals. This suggests that a robust model of discourse structure could complement current robust interpretation systems, which tend to focus on only *one* aspect of the semantically ambiguous material, such as pronouns (e.g., Strübe and Müller (2003)), definite descriptions (e.g., Vieira and Poesio (2000)), or temporal expressions (e.g., Wiebe et al. (1998)). This specialization makes it hard to assess how they would perform in the context of a more comprehensive set of interpretation tasks.

149 PAM: *maybe we can get together, and, discuss, the planning, say, two hours, in the next, couple weeks,*
150 PAM: *let me know what your schedule is like.*
151 CAE: *okay, let me see.*
152 CAE: *twenty,*
153 CAE: *actually, July twenty sixth and twenty seventh looks good,*
154 CAE: *the twenty sixth afternoon,*
155 CAE: *or the twenty seventh, before three p.m., geez.*
156 CAE: *I am out of town the thirtieth through the,*
157 CAE: *the third, I am in San Francisco.*

Figure 1: A dialogue extract from Redwoods.

To date, most methods for constructing discourse structures are not robust. They typically rely on grammatical input and use symbolic methods which inevitably lack coverage. One exception is Marcu's work (Marcu, 1997, 1999) (see also Soricut and Marcu (2003) for constructing discourse structures for individual sentences). Marcu (1999) uses a decision-tree learner and shallow syntactic features

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 96–103, Ann Arbor, June 2005.

to create classifiers for discourse segmentation and for identifying rhetorical relations. Together, these amount to a model of discourse parsing. However, the results are trees of Rhetorical Structure Theory (RST) (Mann and Thompson, 1986), and the classifiers rely on well-formedness constraints on RST trees which are too restrictive (Moore and Pollack, 1992). Furthermore, RST does not offer an account of how compositional semantics gets augmented, nor does it model anaphora. It is also designed for monologue rather than dialogue, so it does not offer a precise semantics of questions or non-sentential utterances which convey propositional content (e.g., 154 and 155 in Figure 1). Another main approach to robust dialogue processing has been statistical models for identifying dialogue acts (e.g., Stolcke et al. (2000)). However, dialogue acts are properties of utterances rather than hierarchically arranged relations *between* them, so they do not provide a basis for resolving semantic underspecification generated by the grammar (Asher and Lascarides, 2003).

Here, we present the first probabilistic approach to parsing the discourse structure of dialogue. We use dialogues from Redwoods' appointment scheduling domain and adapt head-driven generative parsing strategies from sentential parsing (e.g., Collins (2003)) for discourse parsing. The discourse structures we build conform to Segmented Discourse Representation Theory (SDRT) (Asher and Lascarides, 2003). SDRT provides a precise dynamic semantic interpretation for its discourse structures which augments the conventional semantic representations that are built by most grammars. We thus view the task of learning a model of SDRT-style discourse structures as one step towards achieving the goal of robust and precise semantic interpretations.

We describe SDRT in the context of our domain in Section 2. Section 3 discusses how we encode and annotate discourse structures as headed trees for our domain. Section 4 provides background on probabilistic head-driven parsing models, and Section 5 describes how we adapt the approach for discourse and gives four models for discourse parsing. We report results in Section 6, which show the importance of dialogue-based features on performance. Our best model performs far better than a baseline that uses the most frequent rhetorical relations and right-branching segmentation.

h_0 : *Request-Elab*$(149, 150) \wedge$
Plan-Elab$(150, h_1)$
h_1 : *Elaboration*$(153, h_2) \wedge$
Continuation$(153, 156) \wedge$
Continuation$(156, 157)$
h_2 : *Alternation*$(154, 155)$

Figure 2: The SDRS for the dialogue in Figure 1.

2 Segmented Discourse Representation Theory

SDRT extends prior work in dynamic semantics (e.g., van Eijk and Kamp (1997)) via logical forms that feature rhetorical relations. The logical forms consist of *speech act discourse referents* which label content (either of a clause or of text segments). Rhetorical relations such as *Explanation* relate these referents. The resulting structures are called *segmented discourse representation structures* or SDRSs. An SDRS for the dialogue in Figure 1 is given in Figure 2; we have used the numbers of the elementary utterances from Redwoods as the speech act discourse referents but have omitted their labelled logical forms. Note that utterances 151 and 152, which do not contribute to the truth conditions of the dialogue, are absent – we return to this shortly.

There are several things to note about this SDRS. First, SDRT's dynamic semantics of rhetorical relations imposes constraints on the contents of its arguments. For example, *Plan-Elab*$(150, h1)$ (standing for *Plan-Elaboration*) means that $h1$ provides information from which the speaker of 150 can elaborate a plan to achieve their communicative goal (to meet for two hours in the next couple of weeks). The relation *Plan-Elab* contrasts with *Plan-Correction*, which would relate the utterances in dialogue (1):

(1) a. A: *Can we meet at the weekend?*
b. B: *I'm afraid I'm busy then.*

Plan-Correction holds when the content of the second utterance in the relation indicates that its communicative goals conflict with those of the first one. In this case, A indicates he wants to meet next weekend, and B indicates that he does not (note that *then* resolves to the weekend). Utterances (1ab) would also be related with *IQAP* (*Indirect Question Answer*

Pair): this means that (1b) provides sufficient information for the questioner A to infer a direct answer to his question (Asher and Lascarides, 2003).

The relation *Elaboration*$(153, h_2)$ in Figure 2 means that the segment 154 to 155 resolves to a proposition which elaborates part of the content of the proposition 153. Therefore *the twenty sixth* in 154 resolves to the twenty sixth of *July*—any other interpretation contradicts the truth conditional consequences of *Elaboration*. *Alternation*$(154, 155)$ has truth conditions similar to (dynamic) disjunction. *Continuation*$(156, 157)$ means that 156 and 157 have a common topic (here, this amounts to a proposition about when CAE is unavailable to meet).

The second thing to note about Figure 2 is how one rhetorical relation can outscope another: this creates a hierarchical segmentation of the discourse. For example, the second argument to the *Elaboration* relation is the label h_2 of the *Alternation*-segment relating 154 to 155. Due to the semantics of *Elaboration* and *Alternation*, this ensures that the dialogue entails that one of 154 or 155 is true, but it does not entail 154, nor 155.

Finally, observe that SDRT allows for a situation where an utterance connects to more than one subsequent utterance, as shown here with *Elaboration*$(153, h_2) \wedge$ *Continuation*$(153, 156)$. In fact, SDRT also allows two utterances to be related by multiple relations (see (1)) and it allows an utterance to rhetorically connect to multiple utterances in the context. These three features of SDRT capture the fact that an utterance can make more than one illocutionary contribution to the discourse. An example of the latter kind of structure is given in (2):

(2) a. A: *Shall we meet on Wednesday?*
b. A: *How about one pm?*
c. B: *Would one thirty be OK with you?*

The SDRS for this dialogue would feature the relations *Plan-Correction*$(2b, 2c)$, *IQAP*$(2b, 2c)$ and *Q-Elab*$(2a, 2c)$. *Q-Elab*, or *Question-Elaboration*, always takes a question as its second argument; any answers to the question must elaborate a plan to achieve the communicative goal underlying the first argument to the relation. From a logical perspective, recognising *Plan-Correction*$(2b, 2c)$ and *Q-Elab*$(2a, 2c)$ are co-dependent.

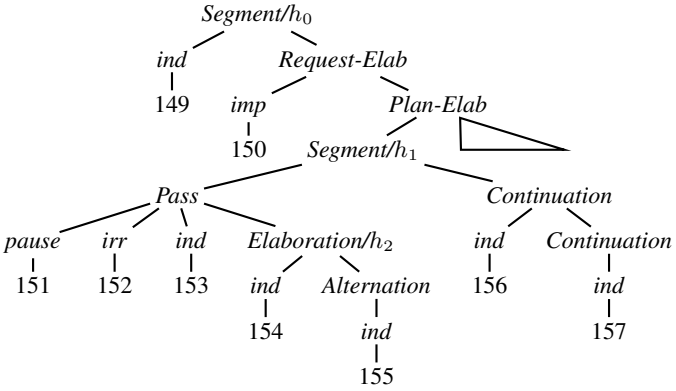


Figure 3: The discourse structure for the dialogue from Figure 1 in tree form.

3 Augmenting the Redwoods treebank with discourse structures

Our starting point is to create training material for probabilistic discourse parsers. For this, we have augmented dialogues from the Redwoods Treebank (Oepen et al., 2002) with their analyses within a fragment of SDRT (Baldridge and Lascarides, 2005). This is a very different effort from that being pursued for the Penn Discourse Treebank (Miltsakaki et al., 2004), which uses discourse connectives rather than abstract rhetorical relations like those in SDRT in order to provide theory neutral annotations. Our goal is instead to leverage the power of the semantics provided by SDRT's relations, and in particular to do so for dialogue as opposed to monologue.

Because the SDRS-representation scheme, as shown in Figure 2, uses graph structures that do not conform to tree constraints, it cannot be combined directly with statistical techniques from sentential parsing. We have therefore designed a headed tree encoding of SDRSs, which can be straightforwardly modeled with standard parsing techniques and from which SDRSs can be recovered.

For instance, the tree for the dialogue in Figure 1 is given in Figure 3. The SDRS in Figure 2 is recovered automatically from it. In this tree, utterances are leaves which are immediately dominated by their tag, indicating either the sentence mood (*ind*icative, *int*errogative or *imp*erative) or that it is *irr*elevant, a *pause* or a pleasantry (e.g., *hello*), annotated as *pls*. Each non-terminal node has a unique head daughter: this is either a *Segment* node, *Pass* node, or a

leaf utterance tagged with its sentence mood. Non-terminal nodes may in addition have any number of daughter *irr*, *pause* and *pls* nodes, and an additional daughter labelled with a rhetorical relation.

The notion of headedness has no status in the semantics of SDRSs themselves. The heads of these discourse trees are not like verbal heads with subcategorization requirements in syntax; here, they are nothing more than the left argument of a rhetorical relation, like 154 in *Alternation*$(154, 155)$. Nonetheless, defining one of the arguments of rhetorical relations as a head serves two main purposes. First, it enables a fully deterministic algorithm for recovering SDRSs from these trees. Second, it is also crucial for creating probabilistic head-driven parsing models for discourse structure.

Segment and *Pass* are non-rhetorical node types. The former explicitly groups multiple utterances. The latter allows its head daughter to enter into relations with segments higher in the tree. This allows us to represent situations where an utterance attaches to more than one subsequent utterance, such as 153 in dialogue (1). Annotators manually annotate the rhetorical relation, *Segment* and *Pass* nodes and determine their daughters. They also tag the individual utterances with one of the three sentence moods or *irr*, *pause* or *pls*. The labels for segments (e.g., h_0 and h_1 in Figure 3) are added automatically. Non-veridical relations such as *Alternation* also introduce segment labels on their parents; e.g., h_2 in Figure 3.

The SDRS is automatically recovered from this tree representation as follows. First, each relation node generates a rhetorical connection in the SDRS: its first argument is the discourse referent of its parent's head daughter, and the second is the discourse referent of the node itself (which unless stated otherwise is its head daughter's discourse referent). For example, the structure in Figure 3 yields *Request-Elab*$(149, 150)$, *Alternation*$(154, 155)$ and *Elaboration*$(153, h_2)$. The labels for the relations in the SDRS—which determine segmentation—must also be recovered. This is easily done: any node which has a segment label introduces an *outscopes* relation between that and the discourse referents of the node's daughters. This produces, for example, *outscopes*$(h_0, 149)$, *outscopes*$(h_1, 153)$ and *outscopes*$(h_2, 154)$. It is straightforward to determine the labels of *all* the rhetorical relations from these conditions. Utterances such as 151 and 152, which are attached with *pause* and *irr* to indicate that they have no overall truth conditional effect on the dialogue, are ignored when constructing the SDRS, so SDRT does not assign these terms any semantics. Overall, this algorithm generates the SDRS in Figure 2 from the tree in Figure 3.

Thus far, 70 dialogues have been annotated and reviewed to create our gold-standard corpus. On average, these dialogues have 237.5 words, 31.5 utterances, and 8.9 speaker turns. In all, there are 30 different rhetorical relations in the inventory for this annotation task, and 6 types of tags for the utterances themselves: *ind*, *int*, *imp*, *pause*, *irr* and *pls*.

Finally, we annotated all 6,000 utterances in the Verbmobil portion of Redwoods with the following: whether the time mentioned (if there is one) is a good time to meet (e.g., *I'm free then* or *Shall we meet at 2pm?*) or a bad time to meet (e.g., *I'm busy then* or *Let's avoid meeting at the weekend*). These are used as features in our model of discourse structure (see Section 5). We use these so as to minimise using directly detailed features from the utterances themselves (e.g. the fact that the utterance contains the word *free* or *busy*, or that it contains a negation), which would lead to sparse data problems given the size of our training corpus. We ultimately aim to *learn* good-time and bad-time from sentence-level features extracted from the 6,000 Redwoods analyses, but we leave this to future work.

4 Generative parsing models

There is a significant body of work on probabilistic parsing, especially that dealing with the English sentences found in the annotated Penn Treebank. One of the most important developments in this work is that of Collins (2003). Collins created several lexicalised head-driven generative parsing models that incorporate varying levels of structural information, such as distance features, the complement/adjunct distinction, subcategorization and gaps. These models are attractive for constructing our discourse trees, which contain heads that establish non-local dependencies in a manner similar to that in syntactic parsing. Also, the co-dependent tasks of determining segmentation and choosing the rhetorical connections are both heavily influenced by the content of

the utterances/segments which are being considered, and lexicalisation allows the model to probabilistically relate such utterances/segments very directly.

Probabilistic Context Free Grammars (PCFGs) determine the conditional probability of a right-hand side of a rule given the left-hand side, $\mathcal{P}(RHS|LHS)$. Collins instead decomposes the calculation of such probabilities by first generating a head and then generating its left and right modifiers independently. In a supervised setting, doing this gathers a much larger set of rules from a set of labelled data than a standard PCFG, which learns only rules that are directly observed.[1]

The decomposition of a rule begins by noting that rules in a lexicalised PCFG have the form:

$$P(h) \rightarrow L_n(l_n) \dots L_1(l_1) H(h) R_1(r_1) \dots R_m(r_m)$$

where h is the head word, $H(h)$ is the label of the head constituent, $P(h)$ is its parent, and $L_i(l_i)$ and $R_i(r_i)$ are the n left and m right modifiers, respectively. It is also necessary to include $STOP$ symbols L_{n+1} and R_{m+1} on either side to allow the Markov process to properly model the sequences of modifiers. By assuming these modifiers are generated independently of each other but are dependent on the head and its parent, the probability of such expansions can be calculated as follows (where $\mathcal{P}_h$, $\mathcal{P}_l$ and $\mathcal{P}_r$ are the probabilities for the head, left-modifiers and right-modifiers respectively):

$$\begin{aligned}
&\mathcal{P}(L_n(l_n) \dots L_1(l_1) H(h) R_1(r_1) \dots R_m(r_m) | P(h)) = \\
&\quad \mathcal{P}_h(H|P(h)) \\
&\quad \times \prod_{i=1\dots n+1} \mathcal{P}_l(L_i(l_i)|P(h), H) \\
&\quad \times \prod_{i=1\dots m+1} \mathcal{P}_r(R_i(r_i)|P(h), H)
\end{aligned}$$

This provides the simplest of models. More conditioning information can of course be added from any structure which has already been generated. For example, Collins' model 1 adds a distance feature that indicates whether the head and modifier it is generating are adjacent and whether a verb is in the string between the head and the modifier.

[1] A similar effect can be achieved by converting n-ary trees to binary form.

5 Discourse parsing models

In Section 3, we outlined how SDRSs can be represented as headed trees. This allows us to create parsing models for discourse that are directly inspired by those described in the previous section. These models are well suited for our discourse parsing task. They are lexicalised, so there is a clear place in the discourse model for incorporating features from utterances: simply replace lexical heads with whole utterances, and exploit features from those utterances in discourse parsing in the same manner as lexical features are used in sentential parsing.

Discourse trees contain a much wider variety of kinds of information than syntactic trees. The leaves of these trees are sentences with full syntactic and semantic analyses, rather than words. Furthermore, each dialogue has two speakers, and speaker style can change dramatically from dialogue to dialogue. Nonetheless, the task is also more constrained in that there are fewer overall constituent labels, there are only a few labels which can act as heads, and trees are essentially binary branching apart from constituents containing ignorable utterances.

The basic features we use are very similar to those for the syntactic parsing model given in the previous section. The feature P is the parent label that is the starting point for generating the head and its modifiers. H is the label of the head constituent. The tag t is also used, except that rather than being a part-of-speech, it is either a sentence mood label (*ind*, *int*, or *imp*) or an ignorable label (*irr*, *pls*, or *pause*). The word feature w in our model is the first discourse cue phrase present in the utterance.[2] In the absence of a cue phrase, w is the empty string. The distance feature Δ is true if the modifier being generated is adjacent to the head and false otherwise. To incorporate a larger context into the conditioning information, we also utilize a feature HCR, which encodes the child relation of a node's head.

We have two features that are particular to dialogue. The first ST, indicates whether the head utterance of a segment starts a turn or not. The other, TC, encodes the number of turn changes within a segment with one of the values 0, 1, or ≥ 2.

Finally, we use the good/bad-time annotations discussed in Section 3 for a feature TM indicating

[2] We obtained our list of cue phrases from Oates (2001).

	Head features							Modifier features								
	P	*t*	*w*	*HCR*	*ST*	*TC*	*TM*	*P*	*t*	*w*	*H*	Δ	*HCR*	*ST*	*TC*	*TM*
Model 1	✓	✓	✓					✓	✓	✓	✓	✓				
Model 2	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓		
Model 3	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	
Model 4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 4: The features active for determining the head and modifier probabilities in each of the four models.

one of the following values for the head utterance of a segment: *good_time*, *bad_time*, *neither*, or *both*.

With these features, we create the four models given in Figure 4. As example feature values, consider the *Segment* node labelled h_1 in Figure 3. Here, the features have as values: P=*Segment*, H=*Pass*, t=*ind* (the tag of utterance 153), w=*Actually* (see 153 in Figure 1), HCR=*Elaboration*, ST=*false*, TC=0, and TM=*good_time*.

As is standard, linear interpolation with back-off levels of decreasing specificity is used for smoothing. Weights for the levels are determined as in Collins (2003).

6 Results

For our experiments, we use a standard chart parsing algorithm with beam search that allows a maximum of 500 edges per cell. The figure of merit for the cut-off combines the probability of an edge with the prior probability of its label, head and head tag. Hypothesized trees that do not conform to some simple discourse tree constraints are also pruned.[3]

The parser is given the elementary discourse units as defined in the corpus. These units correspond directly to the utterances already defined in Redwoods and we can thus easily access their complete syntactic analyses directly from the treebank.

The parser is also given the correct utterance moods to start with. This is akin to getting the correct part-of-speech tags in syntactic parsing. We do this since we are using the parser for semi-automated annotation. Tagging moods for a new discourse is a very quick and reliable task for the human. With them the parser can produce the more complex hierarchical structure more accurately than if it had to guess them – with the potential to dramatically reduce the time to annotate the discourse structures of further dialogues. Later, we will create a sentence mood tagger that presents an n-best list for the parser to start with, from the tag set *ind, int, imp, irr, pause,* and *pls*.

Models are evaluated by using a leave-one-out strategy, in which each dialogue is parsed after training on all the others. We measure labelled and unlabelled performance with both the standard PARSEVAL metric for comparing spans in trees and a relation-based metric that compares the SDRS's produced by the trees. The latter gives a more direct indication of the accuracy of the actual discourse logical form, but we include the former to show performance using a more standard measure. Scores are globally determined rather than averaged over all individual dialogues.

For the relations metric, the relations from the derived discourse tree for the test dialogue are extracted; then, the overlap with relations from the corresponding gold standard tree is measured. For labelled performance, the model is awarded a point for a span or relation which has the correct discourse relation label and both arguments are correct. For unlabelled, only the arguments need to be correct.[4]

Figure 5 provides the f-scores[5] of the various models and compares them against those of a baseline model and annotators. All differences between models are significant, using a pair-wise t-test at 99.5% confidence, except that between the baseline and Model 2 for unlabelled relations.

The baseline model is based on the most frequent way of attaching the current utterance to its dia-

[3] E.g., nodes can have at most one child with a relation label.

[4] This is a much stricter measure than one which measures relations between a head and its dependents in syntax because it requires two *segments* rather than two heads to be related correctly. For example, Model 4's labelled and unlabelled relation f-scores using segments are 43.2% and 67.9%, respectively; on a head-to-head basis, they rise to 50.4% and 81.8%.

[5] The f-score is calculated as $\frac{2 \times precision \times recall}{precision + recall}$.

Model	PARSEVAL Lab.	PARSEVAL Unlab.	Relations Lab.	Relations Unlab.
Baseline	14.7	33.8	7.4	53.3
Model 1	22.7	42.2	23.1	47.0
Model 2	30.1	51.1	31.0	54.3
Model 3	39.4	62.8	39.4	64.4
Model 4	46.3	69.2	43.2	67.9
Inter-annotator	53.7	76.5	50.3	73.0
Annotator-gold	75.9	88.0	75.3	84.0

Figure 5: Model performance.

logue context. The baseline is informed by the gold-standard utterance moods. For this corpus, this results in a baseline which is a right-branching structure, where the relation *Plan-Elaboration* is used if the utterance is indicative, *Question-Elaboration* if it is interrogative, and *Request-Elaboration* if it is imperative. The baseline also appropriately handles ignorable utterances (i.e, those with the mood labels irrelevant, pause, or pleasantry).

The baseline performs poorly on labelled relations (7.4%), but is more competitive on unlabelled ones (53.3%). The main reason for this is that it takes no segmentation risks. It simply relates every non-ignorable utterance to the previous one, which is indeed a typical configuration with common content-level relations like *Continuation*. The generative models take risks that allow them to correctly identify more complex segments – at the cost of missing some of these easier cases.

Considering instead the PARSEVAL scores for the baseline, the labelled performance is much higher (14.7%) and the unlabelled is much lower (33.8%) than for relations. The difference in labelled performance is due to the fact that the intentional-level relations used in the baseline often have arguments that are multi-utterance segments in the gold standard. These are penalized in the relations comparison, but the spans used in PARSEVAL are blind to them. On the other hand, the unlabelled score drops considerably – this is due to poor performance on dialogues whose gold standard analyses do not have a primarily right-branching structure.

Model 1 performs most poorly of all the models. It is significantly better than the baseline on labelled relations, but significantly worse on unlabelled relations. All its features are derived from the structure of the trees, so it gets no clues from speaker turns or the semantic content of utterances.

Model 2 brings turns and larger context via the ST and HCR features, respectively. This improves segmentation over Model 1 considerably, so that the model matches the baseline on unlabelled relations and beats it significantly on labelled relations.

The inclusion of the TC feature in Model 3 brings large (and significant) improvements over Model 2. Essentially, this feature has the effect of penalizing hypothesized content-level segments that span several turns. This leads to better overall segmentation.

Finally, Model 4 incorporates the domain-based TM feature that summarizes some of the semantic content of utterances. This extra information improves the determination of labelled relations. For example, it is especially useful in distinguishing a *Plan-Correction* from a *Plan-Elaboration*.

The overall trend of differences between PARSEVAL and relations scoring show that PARSEVAL is tougher on overall segmentation and relations scoring is tougher on whether a model got the right arguments for each labelled relation. It is the latter that ultimately matters for the discourse structures produced by the parser to be useful; nonetheless, the PARSEVAL scores do show that each model progressively improves on capturing the trees themselves, and that even Model 1 – as a syntactic model – is far superior to the baseline for capturing the overall form of the trees.

We also compare our best model against two upperbounds: (1) inter-annotator agreement on ten dialogues that were annotated independently and (2) the best annotator against the gold standard agreed upon after the independent annotation phase. For the first, the labelled/unlabelled relations f-scores are 50.3%/73.0% and for the latter, they are 75.3%/84.0%—this is similar to the performance on other discourse annotation projects, e.g., Carlson et al. (2001). On the same ten dialogues, Model 4 achieves 42.3%/64.9%.

It is hard to compare these models with Marcu's (1999) rhetorical parsing model. Unlike Marcu, we did not use a variety of corpora, have a smaller training corpus, are analysing dialogues as opposed to monologues, have a larger class of rhetorical relations, and obtain the elementary discourse units

from the Redwoods annotations rather than estimating them. Even so, it is interesting that the scores reported in Marcu (1999) for labelled and unlabelled relations are similar to our scores for Model 4.

7 Conclusion

In this paper, we have shown how the complex task of creating structures for SDRT can be adapted to a standard probabilistic parsing task. This is achieved via a headed tree representation from which SDRSs can be recovered. This enables us to directly apply well-known probabilistic parsing algorithms and use features inspired by them. Our results show that using dialogue-based features are a major factor in improving the performance of the models, both in terms of determining segmentation appropriately and choosing the right relations to connect them.

There is clearly a great deal of room for improvement, even with our best model. Even so, that model performed sufficiently well for use in semi-automated annotation: when correcting the model's output on ten dialogues, one annotator took 30 seconds per utterance, compared to 39 for another annotator working on the same dialogues with no aid.

In future work, we intend to exploit an existing implementation of SDRT's semantics (Schlangen and Lascarides, 2002), which adopts theorem proving to infer resolutions of temporal anaphora and communicative goals from SDRSs for scheduling dialogues. This additional semantic content can in turn be added (semi-automatically) to a training corpus. This will provide further features for learning discourse structure and opportunities for learning anaphora and goal information directly.

Acknowledgments

This work was supported by Edinburgh-Stanford Link R36763, ROSIE project. Thanks to Mirella Lapata and Miles Osborne for comments.

References

N. Asher and A. Lascarides. *Logics of Conversation*. Cambridge University Press, 2003.

J. Baldridge and A. Lascarides. Annotating discourse structures for robust semantic interpretation. In *Proceedings of the 6th International Workshop on Computational Semantics*, Tilburg, The Netherlands, 2005.

L. Carlson, D. Marcu, and M. Okurowski. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the 2nd SIGDIAL Workshop on Discourse and Dialogue, Eurospeech*, 2001.

M. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–638, 2003.

J. R. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63(1–2):69–142, 1993.

W. C. Mann and S. A. Thompson. Rhetorical structure theory: Description and construction of text structures. In G. Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence*, pages 279–300. 1986.

D. Marcu. The rhetorical parsing of unrestricted natural language texts. In *Proceedings of ACL/EACL*, pages 96–103, Somerset, New Jersey, 1997.

D. Marcu. A decision-based approach to rhetorical parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL99)*, pages 365–372, Maryland, 1999.

E. Miltsakaki, R. Prasad, A. Joshi, and B. Webber. The Penn Discourse TreeBank. In *Proceedings of the Language Resources and Evaluation Conference*, Lisbon, Portugal, 2004.

J. D. Moore and M. E. Pollack. A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics*, 18(4):537–544, 1992.

S. Oates. Generating multiple discourse markers in text. Master's thesis, ITRI, University of Brighton, 2001.

S. Oepen, E. Callahan, C. Manning, and K. Toutanova. LinGO Redwoods—a rich and dynamic treebank for HPSG. In *Proceedings of the* LREC *parsing workshop: Beyond* PARSEVAL*, towards improved evaluation measures for parsing systems*, pages 17–22, Las Palmas, 2002.

D. Schlangen and A. Lascarides. Resolving fragments using discourse information. In *Proceedings of the 6th International Workshop on the Semantics and Pragmatics of Dialogue (Edilog)*, Edinburgh, 2002.

R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of Human Language Technology and North American Association for Computational Linguistics*, Edmonton, Canada, 2003.

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, D. Jurafsky R. Bates, P. Taylor, R. Martin, C. van Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–374, 2000.

M. Strübe and C. Müller. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL2003)*, pages 168–175, 2003.

J. van Eijk and H. Kamp. Representing discourse in context. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Linguistics*, pages 179–237. Elsevier, 1997.

R. Vieira and M. Poesio. Processing definite descriptions in corpora. In *Corpus-based and computational approaches to anaphora*. UCL Press, 2000.

J. M. Wiebe, T. P. O'Hara, T. Ohrstrom-Sandgren, and K. J. McKeever. An empirical approach to temporal reference resolution. *Journal of Artificial Intelligence Research*, 9:247–293, 1998.

Intentional Context in Situated Natural Language Learning

Michael Fleischman and Deb Roy
Cognitive Machines
The Media Laboratory
Massachusetts Institute of Technology
mbf@mit.edu, dkroy@media.mit.edu

Abstract

Natural language interfaces designed for situationally embedded domains (e.g. cars, videogames) must incorporate knowledge about the users' context to address the many ambiguities of situated language use. We introduce a model of situated language acquisition that operates in two phases. First, intentional context is represented and inferred from user actions using probabilistic context free grammars. Then, utterances are mapped onto this representation in a noisy channel framework. The acquisition model is trained on unconstrained speech collected from subjects playing an interactive game, and tested on an understanding task.

1 Introduction

As information technologies move off of our desktops and into the world, the need for Natural Language Processing (NLP) systems that exploit information about the environment becomes increasingly apparent. Whether in physical environments (for cars and cell phones) or in virtual ones (for videogames and training simulators), applications are beginning to demand language interfaces that can understand unconstrained speech about constrained domains. Unlike most text-based NLP research, which focuses on open-domain problems, work we refer to as *situated* NLP focuses on improving language processing by exploiting domain-specific information about the non-linguistic situational context of users' interactions. For applications where agents interact in shared environments, such information is critical for successful communication.

Previous work in situated NLP has focused on methods for grounding the meaning of words in physical and virtual environments. The motivation for this work comes from the inability of text-based NLP technologies to offer viable models of semantics for human computer interaction in shared environments. For example, imagine a situation in which a human user is interacting with a robotic arm around a table of different colored objects. If the human were to issue the command "give me the blue one," both the manually-coded (Lenat, 1995; Fellbaum, 1998) and statistical models (Manning and Schutze, 2000) of meaning employed in text-based NLP are inadequate; for, in both models, the meaning of a word is based only on its relations to other words. However, in order for the robot to successfully "give me the blue one," it must be able to link the meaning of the words in the utterance to its perception of the environment (Roy, Hsiao, & Mavridis, 2004). Thus, recent work on grounding meaning has focused on how words and utterances map onto physical descriptions of the environment: either in the form of perceptual representations (Roy, in press, Siskind, 2001, Regier, 1996) or control schemas (Bailey, 1997 Narayanan, 1999).[1]

While such physical descriptions are useful representations for some classes of words (e.g., colors, shapes, physical movements), they are insufficient for more abstract language, such as that which denotes intentional action. This insufficiency stems from the fact that intentional actions (i.e. actions performed with the purpose of achieving a goal) are highly ambiguous when described only in terms of their physically observable characteristics. For example, imagine a situation in which one person moves a cup towards another person and utters the unknown word

[1] Note that Narayanan's work moves away from purely physical to metaphorical levels of description.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 104–111, Ann Arbor, June 2005.

"*blicket*." Now, based only on the physical description of this action, one might come to think of "blicket" as meaning anything from "give cup", to "offer drink", to "ask for change." This ambiguity stems from the lack of contextual information that strictly perceptual descriptions of action provide.

This research presents a methodology for modeling the intentional context of utterances and describes how such representations can be used in a language learning task. We decompose language learning into two phases: intention recognition and linguistic mapping. In the first phase, we model intentional action using a probabilistic context free grammar. We use this model to parse sequences of observed physical actions, thereby inferring a hierarchical tree representation of a user's intentions. In the second phase, we use a noisy channel model to learn a mapping between utterances and nodes in that tree representation. We present pilot situated language acquisition experiments using a dataset of paired spontaneous speech and action collected from human subjects interacting in a shared virtual environment. We evaluate the acquired model on a situated language understanding task.

2 Intention Recognition

The ability to infer the purpose of others' actions has been proposed in the psychological literature as essential for language learning in children (Tommasello, 2003, Regier, 2003). In order to understand how such intention recognition might be modeled in a computational framework, it is useful to specify the types of ambiguities that make intentional actions difficult to model. Using as an example the situation involving the cup described above, we propose that this interaction demonstrates two distinct types of ambiguity. The first type, which we refer to as a *vertical ambiguity* describes the ambiguity between the "move cup" vs. "offer drink" meanings of "blicket." Here the ambiguity is based on the level of description that the speaker intended to convey. Thus, while both meanings are correct (i.e., both meanings accurately describe the action), only one corresponds to the word "blicket."

The second type of ambiguity, referred to as *horizontal ambiguity* describes the ambiguity between the "offer drink" vs. "ask for change" interpretations of "blicket." Here there is an ambiguity based on what actually is the intention behind the physical action. Thus, it is the case that only one of these meaning corresponds to "blicket" *and* the other meaning is not an accurate description of the intended action.

Figure 1 shows a graphical representation of these ambiguities. Here the leaf nodes represent a basic physical description of the action, while the root nodes represent the highest-level actions for which the leaf actions were performed[2]. Such a tree representation is useful in that it shows both the horizontal ambiguity that exists between the nodes labeled "ask for change" and "offer drink," as well as the vertical ambiguity that exits between the nodes labeled "offer drink" and "move cup."

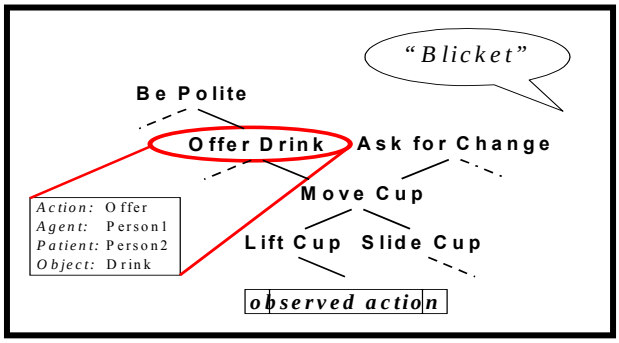


Figure 1: Graphical representation of vertical and horizontal ambiguities for actions.

In order to exploit the intuitive value of such a tree representation, we model intention recognition using probabilistic context free grammars (PCFG)[3]. We develop a small set of production rules in which the left hand side represents a higher order intentional action (e.g., "offer drink"), and the right hand side represents a sequence of lower level actions that accomplish it (e.g. "grasp cup", "move cup", "release cup"). Each individual action (i.e. letter in the alphabet of the PCFG) is further modeled as a simple semantic frame that contains roles for an agent, an object, an action, and multiple optional modifier roles (see inset figure 1). While in this initial work productions are created by hand (a task made feasible by the

[2] In other words, high-level actions (e.g. "be polite) are preformed *by means of* the performance of lower-level actions (e.g. "offer drink").

[3] The idea of a "grammar of behavior" has a rich history in the cognitive sciences dating back at least to Miller et al., 1960

constrained nature of situated domains) learning such rules automatically is discussed in section 4.2.

Just as in the plan recognition work of Pynadath, (1999), we cast the problem of intention recognition as a probabilistic parsing problem in which sequences of physical actions are used to infer an abstract tree representation. Resolving horizontal ambiguities thus becomes equivalent to determining which parse tree is most likely given a sequence of events. Further, resolving vertical ambiguities corresponds to determining which level node in the inferred tree is the correct level of description that the speaker had in mind.

3 Linguistic Mapping

Given a model of intention recognition, the problem for a language learner becomes one of mapping spoken utterances onto appropriate constituents of their inferred intentional representations. Given the intention representation above, this is equivalent to mapping all of the words in an utterance to the role fillers of the appropriate semantic frame in the induced intention tree. To model this mapping procedure, we employ a noisy channel model in which the probability of inferring the correct meaning given an utterance is approximated by the (channel) probability of generating that utterance given that meaning, times the (source) prior probability of the meaning itself (see Equation 1).

$$p(meaning \mid utterance) \approx \quad \textbf{(1)}$$
$$p(utterance \mid meaning)^{\alpha} \bullet p(meaning)^{(1-\alpha)}$$

Here *utterance* refers to some linguistic unit (usually a sentence) and *meaning* refers to some node in the tree (represented as a semantic frame) inferred during intention recognition[4]. We can use the probability associated with the inferred tree (as given by the PCFG parser) as the source probability. Further, we can learn the channel probabilities in an unsupervised manner using a variant of the EM algorithm similar to machine translation (Brown et al., 1993), and statistical language understanding (Epstein, 1996).

4 Pilot Experiments

4.1 Data Collection

In order to avoid the many physical and perceptual problems that complicate work with robots and sensor-grounded data, this work focuses on language learning in virtual environments. We focus on multiplayer videogames , which support rich types of social interactions. The complexities of these environments highlight the problems of ambiguous speech described above, and distinguish this work from projects characterized by more simplified worlds and linguistic interactions, such as SHRDLU (Winograd, 1972). Further, the proliferation of both commercial and military applications (e.g., Rickel et al., 2002) involving such virtual worlds suggests that they will continue to become an increasingly important area for natural language research in the future.

Figure 2: Screen shot of Neverwinter Nights game used in experimentation.

In order to test our model, we developed a virtual environment based on the multi-user videogame Neverwinter Nights.[5] The game, shown in Figure 2, provides useful tools for generating modules in which players can interact. The game was instrumented such that all players' speech/text language and actions are recorded during game play. For data collection, a game was designed in which a single player must navigate their way through a cavernous world, collecting specific objects, in order to escape. Subjects were paired such that one, the *novice*, would control the virtual character, while the other, the *expert,* guided her through the world. While the expert could say anything in order to tell the novice where to go and what to do, the novice was instructed not to speak, but only to follow the commands of the expert.

[4] α refers to a weighting coefficient.

[5] http://nwn.bioware.com/

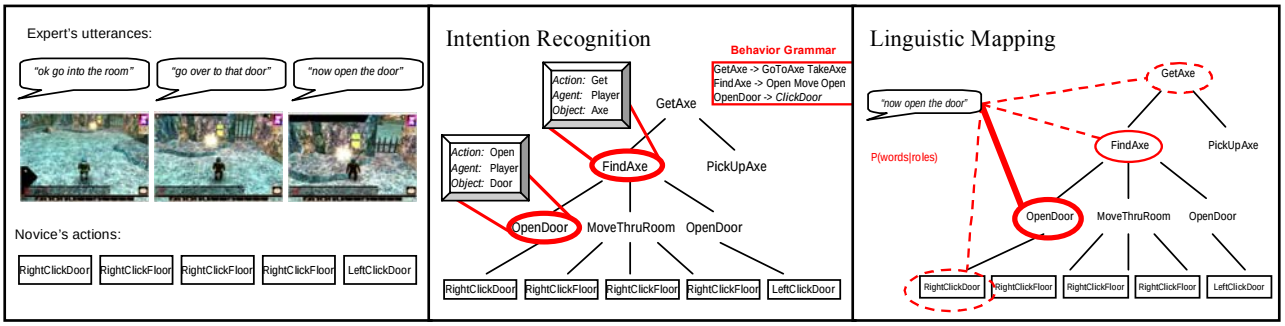


Figure 3. Experimental methodology: a) subjects' speech and action sequences are recorded; b) an intentional tree is inferred over the sequence of observed actions using a PCFG parser; c) the linguistic mapping algorithm examines the mappings between the utterance and all possible nodes to learn the best mapping of words given semantic roles.

The purpose behind these restrictions was to elicit free and spontaneous speech that is only constrained by the nature of the task. This environment seeks to emulate the type of speech that a real situated language system might encounter: i.e., natural in its characteristics, but limited in its domain of discourse.

The subjects in the data collection were university graduate and undergraduate students. Subjects (8 male, 4 female) were staggered such that the novice in one trial became the expert in the next. Each pair played the game at least five times, and for each of those trials, all speech from the expert and all actions from the novice were recorded. Table 1 shows examples of utterances recorded from game play, the observed actions associated with them, and the actions' inferred semantic frame.

Utterance	Action	Frame
ok this time you are gonna get the axe first	`MOVE ROOM1`	*act*: GET *obj*: AXE
through the red archway on your right	`MOVE ROOM2`	act: MOVE *goal*: ARCH *manner*: THRU
now open that door	`CLICK_ON LEVER`	*act*: OPEN *obj*: DOOR
ok now take the axe	`CLICK_ON CHEST`	*act*: TAKE *obj*: AXE *source*: CHEST

Table 1: Representative test utterances collected from subjects with associated game actions and frames

Data collection produces two parallel streams of information: the sequence of actions taken by the novice and the audio stream produced by the expert (figure 3a). The audio streams are automatically segmented into utterances using a speech endpoint detector, which are then transcribed by a human annotator. Each action in the sequence is then automatically parsed, and each node in the tree is replaced with a semantic frame (figure 3b).[6] The data streams are then fed into the linguistic mapping algorithms as a parallel corpus of the expert's transcribed utterances and the inferred semantic roles associated with the novice's actions (figure 3c).

4.2 Algorithms

Intention Recognition

As described in section 2, we represent the task model associated with the game as a set of production rules in which the right hand side consists of an intended action (e.g. "find key") and the left hand side consists of a sequence of sub-actions that are sufficient to complete that action (e.g. "go through door, open chest, pick_up key"). By applying probabilities to the rules, intention recognition can be treated as a probabilistic context free parsing problem, following Pynadath, 1999. For these initial experiments we have hand-annotated the training data in order to generate the grammar used for intention recognition, estimating their maximum likelihood probabilities over the training set. In future work, we intend to examine how such grammars can be learned in conjunction with the language itself; extending research on learning task models (Nicolescu and Mataric, 2003) and work on learning PCFGs (Klein and Manning, 2004) with our own work on unsupervised language learning.

Given the PCFG, we use a probabilistic Earley parser (Stolcke, 1994), modified slightly to output

[6] We use 65 different frames, comprised of 35 unique role fillers.

partial trees (with probabilities) as each action is observed. Figure 4 shows a time slice of an inferred intention tree after a player mouse clicked on a lever in the game. Note that both the vertical and horizontal ambiguities that exist for this action in the game parallel the ambiguities shown in Figure 1. As described above, each node in the tree is represented as a semantic frame (see figure 4 insets), whose roles are aligned to the words in the utterances during the linguistic mapping phase.

Linguistic Mapping

The problem of learning a mapping between linguistic labels and nodes in an inferred intentional tree is recast as one of learning the channel probabilities in Equation 1. Each node in a tree is treated as a simple semantic frame and the role fillers in these frames, along with the words in the utterances, are treated as a parallel corpus. This corpus is used as input to a standard Expectation Maximization algorithm that estimates the probabilities of generating a word given the occurrence of a role filler. We follow IBM Model 1 (Brown et al., 1993) and assume that each word in an utterance is generated by exactly one role in the parallel frame

Using standard EM to learn the role to word mapping is only sufficient if one knows to which level in the tree the utterance should be mapped. However, because of the vertical ambiguity inherent in intentional actions, we do not know in advance which is the correct utterance-to-level mapping. To account for this, we extend the standard EM algorithm as follows (see figure 3c):

1) set uniform likelihoods for all utterance-to-level mappings
2) for each mapping, run standard EM
3) merge output distributions of EM (weighting each by its mapping likelihood)
4) use merged distribution to recalculate likelihoods of all utterance-to-level mappings
5) goto step 2

4.3 Experiments

Methodologies for evaluating language acquisition tasks are not standardized. Given our model, there exists the possibility of employing intrinsic measures of success, such as word alignment accuracy. However, we choose to measure the success of learning by examining the related (and more natural) task of language understanding.

For each subject pair, the linguistic mapping algorithms are trained on the first four trials of game play and tested on the final trial. (This gives on average 130 utterances of training data and 30 utterances of testing data per pair.) For each utterance in the test data, we calculate the likelihood that it was generated by each frame seen in testing. We select the maximum likelihood frame as the system's hypothesized meaning for the test utterance, and examine both how often the maximum likelihood estimate exactly matches the true frame (*frame accuracy*), and how many of the role fillers within the estimated frame match the role fillers of the true frame (*role accuracy*).[7]

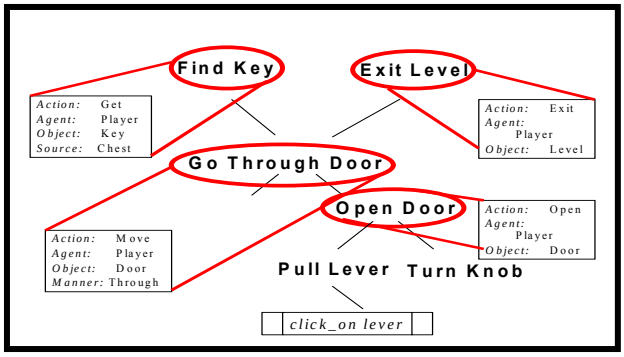


Figure 4: Inferred intention tree (with semantic frames) from human subject game play.

For each subject, the algorithm's parameters are optimized using data from all *other* subjects. We assume correct knowledge of the temporal alignment between utterances and actions. In future work, we will relax this assumption to explore the effects of not knowing which actions correspond to which utterances in time.

To examine the performance of the model, three experiments are presented. Experiment 1 examines the basic performance of the algorithms on the language understanding task described above given uniform priors. The system is tested under two conditions: 1) using the extended EM algorithm given an *unknown* utterance-to-level alignment, and 2) using the standard EM algorithm given the *correct* utterance-to-level alignment.

Experiment 2 tests the benefit of incorporating intentional context directly into language understanding. This is done by using the parse probability of each hypothesized intention as the

[7] See Fleischman and Roy (2005) for experiments detailing performance on specific word categories.

source probability in Equation 1. Thus, given an utterance to understand, we cycle through all possible actions in the grammar, parse each one as if it were observed, and use the probability generated by the parser as its prior probability. By changing the weighting coefficient (α) between the source and channel probabilities, we show the range of performances of the system from using no context at all (α=1) to using only context itself (α=0) in understanding.

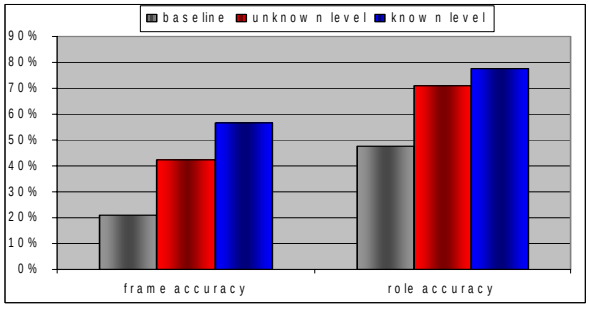


Figure 5: Comparison of models trained with utterance-to-level alignment both known and unknown. Performance is on a language understanding task (baseline equivalent to choosing most frequent frame)

Experiment 3 studies to what extent inferred tree structures are necessary when modeling language acquisition. Although, in section 1, we have presented intuitive reasons why such structures are required, one might argue that inferring trees over sequences of observed actions might not actually improve understanding performance when compared to a model trained only on the observed actions themselves. This hypothesis is tested by comparing a model trained given the correct utterance-to-level alignment (described in experiment 1) with a model in which each utterance is aligned to the leaf node (i.e. observed action) below the correct level of alignment. For example, in figure 4, this would correspond to mapping the utterance "*go through the door*", not to "GO THROUGH DOOR", but rather to "CLICK_ON LEVER."

4.4 Results

Experiment 1: We present the average performance over all subject pairs, trained with the correct utterance-to-level alignment both known and unknown, and compare it to a baseline of choosing the most frequent frame from the training data. Figure 5 shows the percentage of maximum likelihood frames chosen by the system that exactly match the intended frame (frame accuracy), as well as, the percentage of roles from the maximum likelihood frame that overlap with roles in the intended frame (role accuracy).

As expected, the understanding performance goes down for both frames and roles when the correct utterance-to-level alignment is unknown. Interestingly, while the frame performance declines by 14.3%, the performance on roles only declines 6.4%. This difference is due primarily to the fact that, while the mapping from words to action role fillers is hindered by the need to examine all alignments, the mapping from words to object role fillers remains relatively robust. This is due to the fact that while each level of intention carries a different action term, often the objects described at different levels remain the same. For example, in figure 4, the action fillers "TAKE", "MOVE", "OPEN", and "PULL" occur only once along the path. However, the object filler "DOOR" occurs multiple times. Thus, the chance that the role filler "DOOR" correctly maps to the word "*door*" is relatively high compared to the role filler "OPEN" mapping to the word "*open.*"[8]

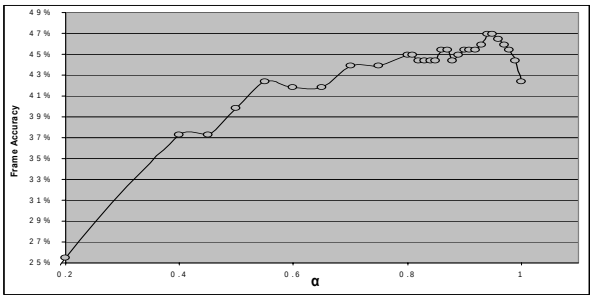


Figure 6: Frame accuracy as a function of α value (Eq. 1) trained on unknown utterance-to-level alignments.

Experiment 2: Figure 6 shows the average frame accuracy of the system trained without knowing the correct utterance-to-level alignment, as a function of varying the α values from Equation 1. The graph shows that including intentional context does improve system performance when it is not given too much weight (i.e., at relatively high alpha values). This suggests that the benefit of intentional context is somewhat outweighed by the power of the learned role to word mappings.

[8] This asymmetry for learning words about actions vs. objects is well known in psychology (Gleitman, 1990) and is addressed directly in Fleischman and Roy, 2005.

Looking closer, we find a strong negative correlation (r=-0.81) between the understanding performance using only channel probabilities (α=1) and the improvement obtained by including the intentional context. In other words, the better one does without context, the less context improves performance. Thus, we expect that in noisier environments (such as when speech recognition is employed) where channel probabilities are less reliable, employing intentional context will be even more advantageous.

Experiment 3: Figure 7 shows the average performance on both frame and role accuracy for systems trained without using the inferred tree structure (on leaf nodes only) and on the full tree structure (given the correct utterance-to-level alignment). Baselines are calculated by choosing the most frequent frame from training.[9]

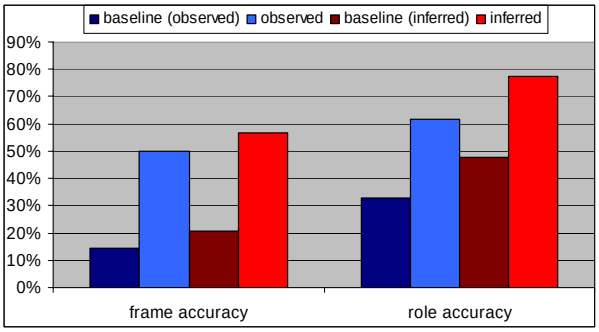


Figure 7: Comparison of models trained on inferred intentional tree vs. directly on observed actions

It is clear from the figure that understanding performance is higher when the intentional tree is used in training. This is a direct result of the fact that speakers often speak about high-level intentions with words that do not directly refer to the observed actions. For example, after opening a door, experts often say: "*go through the door*," for which the observed action is a simple movement (e.g., "MOVE ROOMx"). Also, by referring to high-level intentions, experts can describe sequences of actions that are not immediately referred to. For example, an expert might say: "*get the key*" to describe a sequence of actions that begins with "CLICK_ON CHEST." Thus, the result of not learning over a parsed hierarchical representation of intentions is increased noise, and subsequently, poorer understanding performance.

5 Discussion

The results from these experiments, although preliminary, indicate that this model of language acquisition performs well above baseline on a language understanding task. This is particularly encouraging given the unconstrained nature of the speech on which it was trained. Thus, even free and spontaneous speech can be handled when modeling a constrained domain of discourse.[10]

In addition to performing well given difficult data, the experiments demonstrate the advantages of using an inferred intentional representation both as a contextual aid to understanding and as a representational scaffolding for language learning. More important than these preliminary results, however, is the general lesson that this work suggests about the importance of knowledge representations for situated language acquisition.

As discussed in section 2, learning language about intentional action requires dealing with two distinct types of ambiguity. These difficulties cannot be handled by merely increasing the amount of data used, or switching to a more sophisticated learning algorithm. Rather, dealing with language use for situated applications requires building appropriate knowledge representations that are powerful enough for unconstrained language, yet scalable enough for practical applications. The work presented here is an initial demonstration of how the semantics of unconstrained speech can be modeled by focusing on constrained domains.

As for scalability, it is our contention that for situated NLP, it is not a question of being able to scale up a single model to handle open-domain speech. The complexity of situated communication requires the use of domain-specific knowledge for modeling language use in different contexts. Thus, with situated NLP systems, it is less productive to focus on how to scale up single models to operate beyond their original domains. Rather, as more individual applications are tackled (e.g. cars,

[9] Note that baselines are different for the two conditions, because there are a differing number of frames used in the leaf node only condition.

[10] Notably, situated applications for which natural language interfaces are required typically have limited domains (e.g., talking to one's car doesn't require open-domain language processing).

phones, videogames, etc.) the interesting question becomes one of how agents can learn to switch between different models of language as they interact in different domains of discourse.

6 Conclusion

We have introduced a model of language acquisition that explicitly incorporates intentional contexts in both learning and understanding. We have described pilot experiments on paired language and action data in order to demonstrate both the model's feasibility as well as the efficacy of using intentional context in understanding. Although we have demonstrated a first step toward an advanced model of language acquisition, there is a great deal that has not been addressed. First, what is perhaps most obviously missing is any mention of syntax in the language learning process and its role in bootstrapping for language acquisition. Future work will focus on moving beyond the IBM Model 1 assumptions, to develop more syntactically-structured models.

Further, although the virtual environment used in this research bears similarity to situated applications that demand NL interfaces, it is not known exactly how well the model will perform "in the real world." Future work will examine installing models in real world applications. In parallel investigations, we will explore our method as a cognitive model of human language learning.

Finally, as was mentioned previously, the task model for this domain was hand annotated and, while the constrained nature of the domain simplified this process, further work is required to learn such models jointly with language.

In summary, we have presented first steps toward tackling problems of ambiguity inherent in grounding the semantics of situated language. We believe this work will lead to practical applications for situated NLP, and provide new tools for modeling human cognitive structures and processes underlying situated language use (Fleischman and Roy, 2005).

Acknowledgments

Peter Gorniak developed the software to capture data from the videogame used in our experiments.

References

D. Bailey, J Feldman, S. Narayanan., & G. Lakoff.. Embodied lexical development. 19th Cognitive Science Society Meeting. Mahwah, NJ, 1997.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra & R. L. Mercer. "The Mathematics of Sta-tistical Machine Translation: Parameter Estimation," Computational Linguistics 19(2). 1993.

M Epstein Statistical Source Channel Models for Natural Language Understanding Ph. D. thesis, New York University, September, 1996

C. Fellbaum WordNet: An On-line Lexical Database and Some of its Applications. MIT Press, 1998.

M. Fleischman and D.K. Roy. *Why Verbs are Harder to Learn than Nouns*: Initial Insights from a Computational Model of Intention Recognition in Situated Word Learning. CogSci. Italy, 2005.

L. Gleitman. "The structural sources of verb meanings." Language Acquisition, 1(1), 1990.

D. Klein and C. Manning, "Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency", *Proc. of the 42nd ACL,* 2004

D. B Lenat,. CYC: A Large-Scale Investment in Knowledge Infrastructure". Comm. of ACM, 1995.

C. Manning, H. Schutze,. Foundations of Statistical Natural Language Processing. MIT Press, 2001.

G. A. Miller, E. Galanter, and K. Pribram 1960. Plans and the Structure of Behavior. New York: Halt.

S. Narayanan.. Moving right along: A computational model of metaphoric reasoning about events. In Proc. of AAAI. Orlando, FL, 1999.

M. Nicolescu, M. Mataric´, Natural Methods for Robot Task Learning: Instructive Demonstration, Generalization and Practice, AGENTS, Australia, 2003.

D. Pynadath, 1999. Probabilistic Grammars for Plan Recognition. Ph.D. Thesis, University of Michigan.

T. Regier. The human semantic potential. MIT Press, Cambridge, MA, 1996.

T. Regier. Emergent constraints on word-learning: A computational review. TICS, 7, 263-268, 2003.

J. Rickel, S. Marsella, J. Gratch, R. Hill, D. Traum and W. Swartout, "Towards a New Generation of Virtual Humans for Interactive Experiences," in IEEE Intelligent Systems July/August 2002.

D.Roy, K. Hsiao, and N. Mavridis. Mental imagery for a conversational robot. IEEE Trans. on Systems, Man, and Cybernetics, 34(3) 2004.

D. Roy. (in press). Grounding Language in the World: Schema Theory Meets Semiotics. AI.

J. Siskind. Grounding the Lexical Semantics of Verbs in Visual Perception using Force Dynamics and Event Logic. JAIR, 2001.

A. Stolcke. Bayesian Learning of Probabilistic Language Models. Ph.d., UC Berkeley, 1994.

M. Tomasello. Constructing a Language: A Usage-Based Theory of Language Acquisition. Harvard University Press, 2003.

T. Winograd. *Understanding Natural Language.* Academic Press, 1972.

Representational Bias in Unsupervised Learning of Syllable Structure

Sharon Goldwater and **Mark Johnson**
Department of Cognitive and Linguistic Sciences
Brown University
Providence, RI 02912
{Sharon_Goldwater,Mark_Johnson}@brown.edu

Abstract

Unsupervised learning algorithms based on Expectation Maximization (EM) are often straightforward to implement and provably converge on a local likelihood maximum. However, these algorithms often do not perform well in practice. Common wisdom holds that they yield poor results because they are overly sensitive to initial parameter values and easily get stuck in local (but not global) maxima. We present a series of experiments indicating that for the task of learning syllable structure, the initial parameter weights are not crucial. Rather, it is the choice of model class itself that makes the difference between successful and unsuccessful learning. We use a language-universal rule-based algorithm to find a good set of parameters, and then train the parameter weights using EM. We achieve word accuracy of 95.9% on German and 97.1% on English, as compared to 97.4% and 98.1% respectively for supervised training.

1 Introduction

The use of statistical methods in computational linguistics has produced advances in tasks such as parsing, information retrieval, and machine translation. However, most of the successful work to date has used supervised learning techniques. Unsupervised algorithms that can learn from raw linguistic data, as humans can, remain a challenge. In a statistical framework, one method that can be used for unsupervised learning is to devise a probabilistic model of the data, and then choose the values for the model parameters that maximize the likelihood of the data under the model.

If the model contains hidden variables, there is often no closed-form expression for the maximum likelihood parameter values, and some iterative approximation method must be used. Expectation Maximization (EM) (Neal and Hinton, 1998) is one way to find parameter values that at least locally maximize the likelihood for models with hidden variables. EM is attractive because at each iteration, the likelihood of the data is guaranteed not to decrease. In addition, there are efficient dynamic-programming versions of the EM algorithm for several classes of models that are important in computational linguistics, such as the forward-backward algorithm for training Hidden Markov Models (HMMs) and the inside-outside algorithm for training Probabilistic Context-Free Grammars (PCFGs).

Despite the advantages of maximum likelihood estimation and its implementation via various instantiations of the EM algorithm, it is widely regarded as ineffective for unsupervised language learning. Merialdo (1994) showed that with only a tiny amount of tagged training data, supervised training of an HMM part-of-speech tagger outperformed unsupervised EM training. Later results (e.g. Brill (1995)) seemed to indicate that other methods of unsupervised learning could be more effective (although the work of Banko and Moore (2004) suggests that the difference may be far less than previ-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 112–119, Ann Arbor, June 2005.

ously assumed). Klein and Manning (2001; 2002) recently achieved more encouraging results using an EM-like algorithm to induce syntactic constituent grammars, based on a deficient probability model.

It has been suggested that EM often yield poor results because it is overly sensitive to initial parameter values and tends to converge on likelihood maxima that are local, but not global (Carroll and Charniak, 1992). In this paper, we present a series of experiments indicating that for the task of learning a syllable structure grammar, the initial parameter weights are not crucial. Rather, it is the choice of the model class, i.e., the *representational bias*, that makes the difference between successful and unsuccessful learning.

In the remainder of this paper, we first describe the task itself and the structure of the two different classes of models we experimented with. We then present a deterministic algorithm for choosing a good set of parameters for this task. The algorithm is based on language-universal principles of syllabification, but produces different parameters for each language. We apply this algorithm to English and German data, and describe the results of experiments using EM to learn the parameter weights for the resulting models. We conclude with a discussion of the implications of our experiments.

2 Statistical Parsing of Syllable Structure

Knowledge of syllable structure is important for correct pronunciation of spoken words, since certain phonemes may be pronounced differently depending on their position in the syllable. A number of different supervised machine learning techniques have been applied to the task of automatic syllable boundary detection, including decision-tree classifiers (van den Bosch et al., 1998), weighted finite state transducers (Kiraz and Möbius, 1998), and PCFGs (Müller, 2001; Müller, 2002). The researchers presenting these systems have generally argued from the engineering standpoint that syllable boundary detection is useful for pronunciation of unknown words in text-to-speech systems. Our motivation is a more scientific one: we are interested in the kinds of procedures and representations that can lead to successful unsupervised language learning in both computers and humans.

Our work has some similarity to that of Müller, who trains a PCFG of syllable structure from a corpus of words with syllable boundaries marked. We, too, use a model defined by a grammar to describe syllable structure.[1] However, our work differs from Müller's in that it focuses on how to learn the model's parameters in an unsupervised manner. Several researchers have worked on unsupervised learning of phonotactic constraints and word segmentation (Elman, 2003; Brent, 1999; Venkataraman, 2001), but to our knowledge there is no previously published work on unsupervised learning of syllable structure.

In the work described here, we experimented with two different classes of models of syllable structure. Both of these model classes are presented as PCFGs. The first model class, described in Müller (2002), encodes information about the positions within a word or syllable in which each phoneme is likely to appear. In this *positional* model, each syllable is labeled as initial (I), medial (M), final (F), or as the one syllable in a monosyllabic word (O). Syllables are broken down into an optional onset (the initial consonant or consonant cluster) followed by a rhyme. The rhyme consists of a nucleus (the vowel) followed by an optional coda consonant or cluster. Each phoneme is labeled with a preterminal category of the form *CatPos.x.y*, where *Cat* $\in$ {*Ons, Nuc, Cod*}, *Pos* $\in$ {*I, M, F, O*}, x is the position of a consonant within its cluster, and y is the total number of consonants in the cluster. x and y are unused when *Cat = Nuc*, since all nuclei consist of a single vowel. See Fig. 1 for an example parse.

Rather than directly encoding positional information, the second model class we investigate (the *bigram* model) models statistical dependencies between adjacent phonemes and adjacent syllables. In particular, each onset or coda expands directly into one or more terminal phonemes, thus capturing the ordering dependencies between consonants in a cluster. Also, the shape of each syllable (whether it contains an onset or coda) depends on the shape of the previous syllable, so that the model can learn, for example, that syllables ending in a coda should be followed by syllables with an onset.[2] This kind

[1] We follow Müller in representing our models as PCFGs because this representation is easy to present. The languages generated by these PCFGs are in fact regular, and it is straightforward to transform the PCFGs into equivalent regular grammars.

[2] Many linguists believe that, cross-linguistically, a poten-

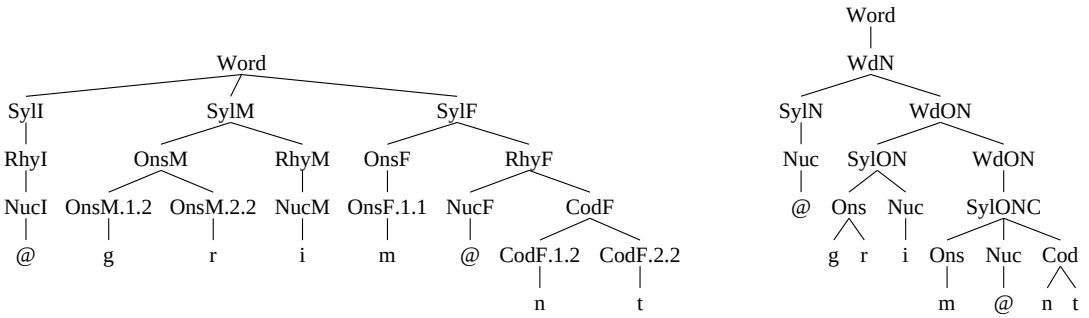


Figure 1: Positional analysis (left) and bigram analysis (right) of the word *agreement*. Groups of terminals dominated by a Syl* node constitute syllables. Terminals appear in the SAMPA encoding of IPA used by CELEX.

of bigram dependency between syllables is modeled using rules of the form WdX $\rightarrow$ Syl$X$ WdY, where X and Y are drawn from the set of possible combinations of onset, nucleus, and coda in a syllable: {N, ON, NC, ONC}. Each Syl$X$ category has only one expansion. See Fig. 1 for an example.

With respect to either of these two model classes, each way of assigning syllable boundaries to a word corresponds to exactly one parse of that word. This makes it simple to train the models from a corpus in which syllable boundaries are provided, as in Müller (2001). We used two different corpora for our experiments, one German (from the ECI corpus of newspaper text) and one English (from the Penn WSJ corpus). Each corpus was created by converting the orthographic forms in the original text into their phonemic transcriptions using the CELEX database (Baayen et al., 1995). CELEX includes syllable boundaries, which we used for supervised training and for evaluation. Any words in the original texts that were not listed in CELEX were discarded, since one of our goals is to compare supervised and unsupervised training.[3] From the resulting phonemic corpora, we created a training set of 20,000 tokens and a test set of 10,000 tokens. Using standard maximum likelihood supervised training procedures, we obtained similar results for models from the two model classes. In German, word accuracy (i.e. the percentage of words with no syllabification errors) was 97.4% for the bigram model and 97.2% for the positional model,[4] while in English it was 98.1% and 97.6% respectively. These results for English are in line with previous reported results using other supervised learning techniques, e.g. van den Bosch et al. (1998). Since many of the words in the data are monosyllabic (49.1% in German, 61.2% in English) and therefore contain no ambiguous syllable boundaries, we also calculated the multisyllabic word accuracy. This was 94.9% (bigram) and 94.5% (positional) in German, and 95.2% (bigram) and 93.8% (positional) in English.

3 Categorical Parsing of Syllable Structure

In the previous section, we described two different model classes and showed that the maximum likelihood estimates with supervised training data yield good models of syllable structure. In moving to unsupervised learning, however, there are two problems that need to be addressed: exactly what class of models do we want to consider (i.e., what kinds of rules should the model contain), and how should we select a particular model from that class (i.e., what weights should the rules have)? We take as our solution to the latter problem the most straightforward approach; namely, maximum likelihood estimation using EM. This leaves us with the question of how to choose a set of parameters in the first place. In this section, we describe an algorithm based on two fundamental phonological principles that, when given a set of data from a particular language, will produce a

tially ambiguous consonant, such as the *b* in *saber*, is always syllabified as the onset of the second syllable rather than the coda of the first. We discuss this point further in Section 3.

[3]Due to the nature of the corpora, the percentage of words discarded was fairly high: 35.6% of the English tokens (primarily proper nouns, acronyms, and numerals, with a smaller number of morphologically complex words) and 26.7% of the German tokens (with compound words making up a somewhat larger portion of these discards).

[4]Müller reports slightly lower results of 96.88% on German using the same positional model. We have no explanation for this discrepancy.

set of rules appropriate to that language. These rules can then be trained using EM.

Given a particular rule schema, it is not immediately clear which of the possible rules should actually be included in the model. For example, in the bigram model, should we start off with the rule *Ons* $\rightarrow$ *k n*? This rule is unnecessary for English, and could lead to incorrect parses of words such as *weakness*. But /kn/ is a legal onset in German, and since we want an algorithm that is prepared to learn any language, disallowing /kn/ as an onset out of hand is unacceptable. On the other hand, the set of all combinatorially possible consonant clusters is infinite, and even limiting ourselves to clusters actually seen in the data for a particular language yields extremely unlikely-sounding onsets like /lkj/ (*calculate*) and /bst/ (*substance*). Ideally, we should limit the set of rules to ones that are likely to actually be used in the language of interest.

The algorithm we have developed for producing a set of language-appropriate rules is essentially a simple categorical (i.e., non-statistical) syllable parser based on the principles of *onset maximization* and *sonority sequencing* (Blevins, 1995). Onset maximization is the idea that in word-medial consonant clusters, as many consonants as possible (given the phonotactics of the language) should be assigned to onset position. This idea is widely accepted and has been codified in Optimality Theory (Prince and Smolensky, 1993) by proposing the existence of a universal preference for syllables with onsets.[5]

In addition to onset maximization, our categorical parser follows the principle of sonority sequencing whenever possible. This principle states that, within a syllable, segments that are closer to the nucleus should be higher in sonority than segments that are further away. Vowels are considered to be the most sonorous segments, followed by glides (/j/, /w/), liquids (/l/, /r/), nasals (/n/, /m/, /ŋ/), fricatives (/v/, /s/, /θ/, ...), and stops (/b/, /t/, /k/, ...). Given a cluster of consonants between two syllable nuclei, sonority sequencing states that the syllable boundary should occur either just before or just after the consonant with lowest sonority. Combining this principle with onset maximization predicts that the boundary should fall before the lowest-sonority segment.

Predicting syllable boundaries in this way is not foolproof. In some cases, clusters that are predicted by sonority sequencing to be acceptable are in fact illegal in some languages. The illegal English onset cluster *kn* is a good example. In other cases, such as the English onset *str*, clusters are allowed despite violating sonority sequencing. These mismatches between universal principles and language-specific phonotactics lead to errors in the predictions of the categorical parser, such as *wea.kness* and *ins.tru.ment*. In addition, certain consonant clusters like *bst* (as in *substance*) may contain more than one minimum sonority point. To handle these cases, the categorical parser follows onset maximization by adding any consonants occurring between the two minima to the onset of the second syllable: *sub.stance*.

Not surprisingly, the categorical parser does not perform as well as the supervised statistical parser: only 92.7% of German words and 94.9% of English words (85.7% and 86.8%, respectively, of multisyllabic words) are syllabified correctly. However, a more important result of parsing the corpus using the categorical parser is that its output can be used to define a model class (i.e., a set of PCFG rules) from which a model can be learned using EM.

Specifically, our model class contains the set of rules that were proposed at least once by the categorical parser in its analysis of the training corpus; in the EM experiments described below, the rule probabilities are initialized to their frequency in the categorical parser's output. Due to the mistakes made by the categorical parser, there will be some rules, like *Ons* $\rightarrow$ *k n* in English, that are not present in the model trained on the true syllabification, but many possible but spurious rules, such as *Ons* $\rightarrow$ *b s t*, will be avoided. Although clusters that violate sonority sequencing tend to be avoided by the categorical parser, it does find examples of these types of clusters at the beginnings and endings of words, as well as occasionally word-medially (as in *sub.stance*). This means that many legal clusters that

[5] An important point, which we return to in Section 5, is that exceptions to onset maximization may occur at morpheme boundaries. Some linguists also believe that there are additional exceptions in certain languages (including English and German), where stressed syllables attract codas. Under this theory, the correct syllabification for *saber* would not be *sa.ber*, but rather *sab.er*, or possibly *sa[b]er*, where the *[b]* is ambisyllabic. Since the syllable annotations in the CELEX database follow simple onset maximization, we take that as our approach as well and do not consider stress when assigning syllable boundaries.

	Bigram		Positional	
	all	multi	all	multi
CP	92.7	85.7	92.7	85.7
CP + EM	95.9	91.9	91.8	84.0
CP-U + EM	95.9	91.9	92.0	84.4
supervised	97.4	94.9	97.2	94.5
SP + EM	71.6	44.3	94.4	89.1
SP-U + EM	71.6	44.3	94.4	89.0

Table 1: Results for German: % of all words (or multisyllabic words) correctly syllabified.

	Bigram		Positional	
	all	multi	all	multi
CP	94.9	86.8	94.9	86.8
CP + EM	97.1	92.6	94.1	84.9
CP-U + EM	97.1	92.6	94.1	84.9
supervised	98.1	95.2	97.6	93.8
SP + EM	86.0	64.0	96.5	90.9
SP-U + EM	86.0	64.0	67.6	16.5

Table 2: Results for English.

violate sonority sequencing will also be included in the set of rules found by this procedure, although their probabilities may be considerably lower than those of the supervised model. In the following section, we show that these differences in rule probabilities are unimportant; in fact, it is not the rule probabilities estimated from the categorical parser's output, but only the set of rules itself that matters for successful task performance.

4 Experiments

In this section, we present a series of experiments using EM to learn a model of syllable structure. All of our experiments use the same German and English 20,000-word training corpora and 10,000-word testing corpora as described in Section 2.[6]

For our first experiment, we ran the categorical parser on the training corpora and estimated a model from the parse trees it produced, as described in the previous section. This is essentially a single step of Viterbi EM training. We then continued to train the model by running (standard) EM to convergence. Results of this experiment with Categorical Parsing + EM (CP + EM) are shown in Tables 1 and 2. For both German and English, using this learning method with the bigram model yields performance that is much better than the categorical parser alone, though not quite as good as the fully supervised regime. On the other hand, training a positional model from the categorical parser's output and then running EM causes performance to degrade.

To determine whether the good performance of the bigram model was simply due to good initialization of the parameter weights, we performed a second experiment. Again starting with the set of rules output by the categorical parser, we initialized the rule weights to the uniform distribution. The results of this experiment (CP-U + EM) show that for the class of bigram models, the performance of the final model found by EM does not depend on the initial rule probabilities. Performance within the positional model framework does depend on the initial rule probabilities, since accuracy in German is different for the two experiments.

As we have pointed out, the rules found by the categorical parser are not exactly the same as the rules found using supervised training. This raises the question of whether the difference in performance between the unsupervised and supervised bigram models is due to differences in the rules. To address this question, we performed two additional experiments. First, we simply ran EM starting from the model estimated from supervised training data. Second, we kept the set of rules from the supervised training data, but reinitialized the probabilities to a uniform distribution before running EM. The results of these experiments are shown as SP + EM and SP-U + EM, respectively. Again, performance of the bigram model is invariant with respect to initial parameter values, while the performance of the positional model is not. Interestingly, the performance of the bigram model in these two experiments is far worse than in the CP experiments. This result is counterintuitive, since it would seem that the model rules found by the supervised system are the optimal rules for this task. In the following section, we explain why these rules are not, in fact, the optimal rules for unsupervised learning, as well as why we believe the bigram model performs so much better

[6] Of course, for unsupervised learning, it is not necessary to use a distinct testing corpus. We did so in order to use the same testing corpus for both supervised and unsupervised learning experiments, to ensure fair comparison of results.

than the positional model in the unsupervised learning situation.

5 Discussion

The results of our experiments raise two interesting questions. First, when starting from the categorical parser's output, why does the bigram model improve after EM training, while the positional model does not? And second, why does applying EM to the supervised bigram model lead to worse performance than applying it to the model induced from the categorical parser?

To answer the first question, notice that one difference between the bigram model and the positional model is that onsets and codas in the bigram model are modeled using the same set of parameters regardless of where in the word they occur. This means that the bigram model generalizes whatever it learns about clusters at word edges to word-medial clusters (and, of course, vice versa). Since the categorical parser only makes errors word-medially, incorrect clusters are only a small percentage of clusters overall, and the bigram model can overcome these errors by reanalyzing the word-medial clusters. The errors that are made after EM training are mostly due to overgeneralization from clusters that are very common at word edges, e.g. predicting le.gi.sla.tion instead of le.gis.la.tion.

In contrast to the bigram model, the positional model does not generalize over different positions of the word, which means that it learns and repeats the word-medial errors of the categorical parser. For example, this model predicts /ɛ.gzɛ.kju.tɪv/ for *executive*, just as the categorical parser does, although /gz/ is never attested in word-initial position. In addition, each segment in a cluster is generated independently, which means clusters like /tl/ may be placed together in an onset because /t/ is common as the first segment of an onset, and /l/ is common as the second. While this problem exists even in the supervised positional model, it is compounded in the unsupervised version because of the errors of the categorical parser.

The differences between these two models are an example of the bias-variance trade-off in probabilistic modeling (Geman et al., 1992): models with low bias will be able to fit a broad range of observations fairly closely, but slight changes in the observed data will cause relatively large changes in the induced model. On the other hand, models with high bias are less sensitive to changes in the observed data. Here, the bigram model induced from the categorical parser has a relatively high bias: regardless of the parameter weights, it will be a poor model of data where word-medial onsets and codas are very different from those at word edges, and it cannot model data with certain onsets such as /vp/ or /tz/ at all because the rules $Ons \rightarrow v\ p$ and $Ons \rightarrow t\ z$ are simply absent. The induced positional model can model both of these situations, and can fit the true parses more closely as well (as evidenced by the fact that the likelihood of the data under the supervised positional model is higher than the likelihood under the supervised bigram model). As a result, however, it is more sensitive to the initial parameter weights and learns to recreate the errors produced by the categorical parser. This sensitivity to initial parameter weights also explains the extremely poor performance of the positional model in the SP-U + EM experiment on English. Because the model is so unconstrained, in this case it finds a completely different local maximum (not the global maximum) which more or less follows coda maximization rather than onset maximization, yielding syllabifications like *synd.ic.ate* and *tent.at.ive.ly*.

The concept of representational bias can also explain why applying EM to the supervised bigram model performs so poorly. Examining the model induced from the categorical parser reveals that, not surprisingly, it contains more rules than the supervised bigram model. This is because the categorical parser produces a wider range of onsets and codas than there are in the true parses. However, the induced model is not a superset of the supervised model. There are four rules (three in English) that occur in the supervised model but not the induced model. These are the rules that allow words where one syllable contains a coda and the following syllable has no onset. These are never produced by the categorical parser because of its onset-maximization principle. However, it turns out that a very small percentage of words do follow this pattern (about .14% of English tokens and 1.1% of German tokens). In English, these examples seem to consist entirely of words where the unusual syllable boundary occurs at a morpheme boundary (e.g. *un.usually, dis.appoint,*

week.end, turn.over). In German, all but a handful of examples occur at morpheme boundaries as well.[7]

The fact that the induced bigram model is unable to model words with codas followed by no onset is a very strong bias, but these words are so infrequent that the model can still fit the data quite well. The missing rules have no effect on the accuracy of the parser, because in the supervised model the probabilities on the rules allowing these kinds of words are so low that they are never used in the Viterbi parses anyway. The problem is that if these rules are included in the model prior to running EM, they add several extra free parameters, and suddenly EM is able to reanalyze many of the words in the corpus to make better use of these parameters. It ends up preferring certain segments and clusters as onsets and others as codas, which raises the likelihood of the corpus but leads to very poor performance. Essentially, it seems that the presence of a certain kind of morpheme boundary is an additional parameter of the "true" model that the bigram model doesn't include. Trying to account for the few cases where this parameter matters requires introducing extra parameters that allow EM too much freedom of analysis. It is far better to constrain the model, disallowing certain rare analyses but enabling the model to learn successfully in a way that is robust to variations in initial conditions and idiosyncracies of the data.

6 Conclusion

We make no claims that our learning system embodies a complete model of syllabification. A full model would need to account for the effects of morphological boundaries, as well as the fact that some languages allow resyllabification over word boundaries. Nevertheless, we feel that the results presented here are significant. We have shown that, despite previous discouraging results (Carroll and Charniak, 1992; Merialdo, 1994), it is possible to achieve good results using EM to learn linguistic structures in an unsupervised way. However, the choice of model parameters is crucial for successful learning. Carroll and Charniak, for example, generated all possible rules within a particular framework and relied on EM to remove the "unnecessary" rules by letting their probabilities go to zero. We suggest that this procedure tends to yield models with low bias but high variance, so that they are extremely sensitive to the small variations in expected rule counts that occur with different initialization weights.

Our work suggests that using models with higher bias but lower variance may lead to much more successful results. In particular, we used universal phonological principles to induce a set of rules within a carefully chosen grammatical framework. We found that there were several factors that enabled our induced bigram model to learn successfully where the comparison positional model did not:

1. The bigram model encodes bigram dependencies of syllable shape and disallows onset-less syllables following syllables with codas.

2. The bigram model does not distinguish between different positions in a word, so it can generalize onset and coda sequences from word edges to word-medial position.

3. The bigram model learns specific sequences of legal clusters rather than information about which positions segments are likely to occur in.

Notice that each of these factors imposes a constraint on the kinds of data that can be modeled. We have already discussed the fact that item 1 rules out the correct syllabification of certain morphologically complex words, but since our system currently has no way to determine morpheme boundaries, it is better to do so than to introduce extra free parameters. One possible extension to this work would be to try to incorporate morphological boundary information (either annotated or induced) into the model.

A more interesting constraint is the one imposed by item 2, since in fact most languages do have some differences between the onsets and (especially) codas allowed at word edges and within words. However, the proper way to handle this fact is not by introducing completely independent parameters for initial, medial, and final positions, since this allows far too much freedom. It would be extremely surprising to find a language with one set of codas allowed word-internally, and a completely disjoint set

[7] The exceptions in our training data were *auserkoren 'chosen'*, *erobern 'capture'*, and forms of *erinnern 'remind'*, all of which were listed in CELEX as having a syllable boundary, but no morpheme boundary, after the first consonant. Our knowledge of German is not sufficient to determine whether there is some other factor that can explain these cases.

allowed word-finally. In fact, the usual situation is that word-internal onsets and codas are a subset of those allowed at word edges, and this is exactly why using word edges to induce our rules was successful.

Considering language more broadly, it is common to find patterns of linguistic phenomena with many similarities but some differences as well. For such cases, adding extra parameters to a supervised model often yields better performance, since the augmented model can capture both primary and secondary effects. But it seems that, at least for the current state of unsupervised learning, it is better to limit the number of parameters and focus on those that capture the main effects in the data. In our task of learning syllable structure, we were able to use just a few simple principles to constrain the model successfully. For more complex tasks such as syntactic parsing, the space of linguistically plausible models is much larger. We feel that a research program integrating results from the study of linguistic universals, human language acquisition, and computational modeling is likely to yield the most insight into the kinds of constraints that are needed for successful learning.

Ultimately, of course, we will want to be able to capture not only the main effects in the data, but some of the subtler effects as well. However, we believe that the way to do this is not by introducing completely free parameters, but by using a Bayesian prior that would enforce a degree of similarity between certain parameters. In the meantime, we have shown that employing linguistic universals to determine which set of parameters to include in a language model for syllable parsing allows us to use EM for learning the parameter weights in a successful and robust way.

Acknowledgments

We would like to thank Eugene Charniak and our colleagues in BLLIP for their support and helpful suggestions. This research was partially supported by NSF awards IGERT 9870676 and ITR 0085940 and NIMH award 1R0-IMH60922-01A2.

References

R. Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX lexical database (release 2) [cd-rom].

M. Banko and R. Moore. 2004. A study of unsupervised part-of-speech tagging. In *Proceedings of COLING '04*.

J. Blevins. 1995. The syllable in phonological theory. In J. Goldsmith, editor, *the Handbook of Phonological Theory*. Blackwell, Oxford.

M. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

E. Brill. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the 3rd Workshop on Very Large Corpora*, pages 1–13.

G. Carroll and E. Charniak. 1992. Two experiments on learning probabilistic dependency grammars from corpora. In *Proceedings of the AAAI Workshop on Statistically-Based Natural Language Processing Techniques*, San Jose, CA.

J. Elman. 2003. Generalization from sparse input. In *Proceedings of the 38th Annual Meeting of the Chicago Linguistic Society*.

S. Geman, E. Bienenstock, and R. Doursat. 1992. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58.

G. A. Kiraz and B. Möbius. 1998. Multilingual syllabification using weighted finite-state transducers. In *Proceedings of the Third European Speech Communication Association Workshop on Speech Synthesis*.

D. Klein and C. Manning. 2001. Distributional phrase structure induction. In *Proceedings of the Conference on Natural Language Learning*, pages 113–120.

D. Klein and C. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the ACL*.

B. Merialdo. 1994. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

K. Müller. 2001. Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of the ACL*.

K. Müller. 2002. Probabilistic context-free grammars for phonology. In *Proceedings of the Workshop on Morphological and Phonological Learning at ACL*.

R. Neal and G. Hinton, 1998. *A New View of the EM Algorithm That Justifies Incremental and Other Variants*, pages 355–368. Kluwer.

A. Prince and P. Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical Report TR-2, Rutgers Center for Cognitive Science, Rutgers Univ.

A. van den Bosch, T. Weijters, and W. Daelemans. 1998. Modularity in inductively-learned word pronunciation systems. In *New Methods in Language Processing and Computational Language Learning (NeMLaP3/CoNLL98)*.

A. Venkataraman. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):351–372.

An Analogical Learner for Morphological Analysis

Nicolas Stroppa & François Yvon
GET/ENST & LTCI, UMR 5141
46 rue Barrault, 75013 Paris, France
{stroppa,yvon}@enst.fr

Abstract

Analogical learning is based on a two-step inference process: (i) computation of a structural mapping between a new and a memorized situation; (ii) transfer of knowledge from the known to the unknown situation. This approach requires the ability to search for and exploit such mappings, hence the need to properly define analogical relationships, and to efficiently implement their computation.

In this paper, we propose a unified definition for the notion of (formal) analogical proportion, which applies to a wide range of algebraic structures. We show that this definition is suitable for learning in domains involving large databases of structured data, as is especially the case in Natural Language Processing (NLP). We then present experimental results obtained on two morphological analysis tasks which demonstrate the flexibility and accuracy of this approach.

1 Introduction

Analogical learning (Gentner et al., 2001) is based on a two-step inductive process. The first step consists in the construction of a *structural* mapping between a new instance of a problem and solved instances of the same problem. Once this mapping is established, solutions for the new instance can be induced, based on one or several analogs. The implementation of this kind of inference process requires techniques for searching for, and reasoning with, structural mappings, hence the need to properly define the notion of analogical relationships and to efficiently implement their computation.

In Natural Language Processing (NLP), the typical dimensionality of databases, which are made up of hundreds of thousands of instances, makes the search for complex structural mappings a very challenging task. It is however possible to take advantage of the specific nature of linguistic data to work around this problem. Formal (surface) analogical relationships between linguistic representations are often a good sign of deeper analogies: a surface similarity between the word strings *write* and *writer* denotes a deeper (semantic) similarity between the related concepts. Surface similarities can of course be misleading. In order to minimize such confusions, one can take advantage of other specificities of linguistic data: (i) their systemic organization in (pseudo)-paradigms, and (ii) their high level of redundancy. In a large lexicon, we can indeed expect to find many instances of pairs like *write-writer*: for instance *read-reader*, *review-reviewer*...

Complementing surface analogies with statistical information thus has the potential to make the search problem tractable, while still providing with many good analogs. Various attempts have been made to use surface analogies in various contexts: automatic word pronunciation (Yvon, 1999), morphological analysis (Lepage, 1999a; Pirrelli and Yvon, 1999) and syntactical analysis (Lepage, 1999b). These experiments have mainly focused on linear represen-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 120–127, Ann Arbor, June 2005.

tations of linguistic data, taking the form of finite sequences of symbols, using a restrictive and sometimes *ad-hoc* definition of the notion of an analogy.

The first contribution of this paper is to propose a general definition of formal analogical proportions for algebraic structures commonly used in NLP: attribute-value vectors, words on finite alphabets and labeled trees. The second contribution is to show how these formal definitions can be used within an instance-based learning framework to learn morphological regularities.

This paper is organized as follows. In Section 2, our interpretation of analogical learning is introduced and related to other models of analogical learning and reasoning. Section 3 presents a general algebraic framework for defining analogical proportions as well as its instantiation to the case of words and labeled trees. This section also discusses the algorithmic complexity of the inference procedure. Section 4 reports the results of experiments aimed at demonstrating the flexibility of this model and at assessing its generalization performance. We conclude by discussing current limitations of this model and by suggesting possible extensions.

2 Principles of analogical learning

2.1 Analogical reasoning

The ability to identify analogical relationships between what looks like unrelated situations, and to use these relationships to solve complex problems, lies at the core of human cognition (Gentner et al., 2001). A number of models of this ability have been proposed, based on symbolic (e.g. (Falkenheimer and Gentner, 1986; Thagard et al., 1990; Hofstadter and the Fluid Analogies Research group, 1995)) or subsymbolic (e.g. (Plate, 2000; Holyoak and Hummel, 2001)) approaches. The main focus of these models is the dynamic process of analogy making, which involves the identification of a structural mappings between a memorized and a new situation. Structural mapping relates situations which, while being apparently very different, share a set of common high-level relationships. The building of a structural mapping between two situations utilizes several subparts of their descriptions and the relationships between them.

Analogy-making seems to play a central role in our reasoning ability; it is also invoked to explain some human skills which do not involve any sort of conscious reasoning. This is the case for many tasks related to the perception and production of language: lexical access, morphological parsing, word pronunciation, etc. In this context, analogical models have been proposed as a viable alternative to rule-based models, and many implementation of these low-level analogical processes have been proposed such as decision trees, neural networks or instance-based learning methods (see e.g. (Skousen, 1989; Daelemans et al., 1999)). These models share an acceptation of analogy which mainly relies on surface *similarities* between instances.

Our learner tries to bridge the gap between these approaches and attempts to remain faithful to the idea of structural analogies, which prevails in the AI literature, while also exploiting the intuitions of large-scale, instance-based learning models.

2.2 Analogical learning

We consider the following supervised learning task: a learner is given a set $\mathcal{S}$ of training instances $\{X_1, \ldots, X_n\}$ independently drawn from some unknown distribution. Each instance X_i is a vector containing m features: $\langle X_{i1}, \ldots, X_{im} \rangle$. Given $\mathcal{S}$, the task is to predict the missing features of partially informed new instances. Put in more standard terms, the set of known (resp. unknown) features for a new value X forms the *input space* (resp. *output space*): the projections of X onto the input (resp. output) space will be denoted $I(X)$ (resp. $O(X)$). This setting is more general than the simpler classification task, in which only one feature (the class label) is unknown, and covers many other interesting tasks.

The inference procedure can be sketched as follows: training examples are simply stored for future use; no generalization (abstraction) of the data is performed, which is characteristic of *lazy learning* (Aha, 1997). Given a new instance X, we identify formal analogical proportions involving X in the input space; known objects involved in these proportions are then used to infer the missing features.

An analogical proportion is a relation involving four objects A, B, C and D, denoted by $A : B :: C : D$ and which reads A *is to* B *as* C *is to* D. The definition and computation of these proportions are studied in Section 3. For the moment,

we contend that it is possible to construct analogical proportions between (possibly partially informed) objects in $\mathcal{S}$. Let $I(X)$ be a partially described object not seen during training. The analogical inference process is formalized as:

1. Construct the set $\mathcal{T}(X) \subset \mathcal{S}^3$ defined as:

$$\mathcal{T}(X) = \{(A, B, C) \in \mathcal{S}^3 \mid I(A) : I(B) :: I(C) : I(X)\}$$

2. For each $(A, B, C) \in \mathcal{T}(X)$, compute hypotheses $\widehat{O(X)}$ by solving the equation:

$$\widehat{O(X)} = O(A) : O(B) :: O(C) :?$$

This inference procedure shows lots of similarities with the k-nearest neighbors classifier (k-NN) which, given a new instance, (i) searches the training set for close neighbors, (ii) compute the unknown class label according to the neighbors' labels. Our model, however, does not use any metric between objects: we only rely on the definition of analogical proportions, which reveal systemic, rather than superficial, similarities. Moreover, inputs and outputs are regarded in a symmetrical way: outputs are not restricted to a set of labels, and can also be structured objects such as sequences. The implementation of the model still has to address two specific issues.

- When exploring $\mathcal{S}^3$, an exhaustive search evaluates $|\mathcal{S}|^3$ triples, which can prove to be intractable. Moreover, objects in $\mathcal{S}$ may be unequally relevant, and we might expect the search procedure to treat them accordingly.
- Whenever several competing hypotheses are proposed for $\widehat{O(X)}$, a ranking must be performed. In our current implementation, hypotheses are ranked based on frequency counts.

These issues are well-known problems for k-NN classifiers. The second one does not appear to be critical and is usually solved based on a majority rule. In contrast, a considerable amount of effort has been devoted to reduce and optimize the search process, via editing and condensing methods, as studied e.g. in (Dasarathy, 1990; Wilson and Martinez, 2000). Proposals for solving this problem are discussed in Section 3.4.

3 An algebraic framework for analogical proportions

Our inductive model requires the availability of a device for computing analogical proportions on feature vectors. We consider that an analogical proportion holds between four feature vectors when the proportion holds for all components. In this section, we propose a unified algebraic framework for defining analogical proportions between individual features. After giving the general definition, we present its instantiation for two types of features: words over a finite alphabet and sets of labelled trees.

3.1 Analogical proportions

Our starting point will be analogical proportions in a set U, which we define as follows: $\forall x, y, z, t \in U, x : y :: z : t$ if and only if either $x = y$ and $z = t$ or $x = z$ and $y = t$. In the sequel, we assume that U is additionally provided with an associative internal composition law $\oplus$, which makes $(U, \oplus)$ a semigroup. The generalization of proportions to semigroups involves two key ideas: the *decomposition* of objects into smaller parts, subject to *alternation constraints*. To formalize the idea of decomposition, we define the *factorization* of an element u in U as:

Definition 1 (Factorization)
A factorization *of $u \in U$ is a sequence $u_1 \dots u_n$, with $\forall i, u_i \in U$, such that: $u_1 \oplus \dots \oplus u_n = u$. Each term u_i is a* factor *of u.*

The alternation constraint expresses the fact that analogically related objects should be made of alternating factors: for $x : y :: z : t$ to hold, each factor in x should be found alternatively in y and in z. This yields a first definition of analogical proportions:

Definition 2 (Analogical proportion)
$(x, y, z, t) \in U$ form an analogical proportion*, denoted by $x : y :: z : t$ if and only if there exists some factorizations $x_1 \oplus \dots \oplus x_d = x$, $y_1 \oplus \dots \oplus y_d = y$, $z_1 \oplus \dots \oplus z_d = z$, $t_1 \oplus \dots \oplus t_d = t$ such that $\forall i, (y_i, z_i) \in \{(x_i, t_i), (t_i, x_i)\}$. The smallest d for which such factorizations exist is termed the* degree *of the analogical proportion.*

This definition is valid for any semigroup, and *a fortiori* for any richer algebraic structure. Thus, it readily applies to the case of groups, vector spaces, free monoids, sets and attribute-value structures.

3.2 Words over Finite Alphabets

3.2.1 Analogical Proportions between Words

Let Σ be a finite alphabet. $\Sigma^\star$ denotes the set of finite sequences of elements of Σ, called *words* over Σ. $\Sigma^\star$, provided with the concatenation operation . is a free monoid whose identity element is the empty word ε. For $w \in \Sigma^\star$, $w(i)$ denotes the i^{th} symbol in w. In this context, definition (2) can be re-stated as:

Definition 3 (Analogical proportion in ($\Sigma^\star$,.)) *$(x, y, z, t) \in \Sigma^\star$ form an analogical proportion, denoted by $x : y :: z : t$ if and only if there exists some integer d and some factorizations $x_1 \dots x_d = x$, $y_1 \dots y_d = y$, $z_1 \dots z_d = z$, $t_1 \dots t_d = t$ such that $\forall i, (y_i, z_i) \in \{(x_i, t_i), (t_i, x_i)\}$.*

An example of analogy between words is:

$$viewing : reviewer :: searching : researcher$$

with $x_1 = \epsilon$, $x_2 = view$, $x_3 = ing$ and $t_1 = re$, $t_2 = search$, $t_3 = er$. This definition generalizes the proposal of (Lepage, 1998). It does not ensure the existence of a solution to an analogical equation, nor its uniqueness when it exists. (Lepage, 1998) gives a set of necessary conditions for a solution to exist. These conditions also apply here. In particular, if t is a solution of $x : y :: z :?$, then t contains, in the same relative order, all the symbols in y and z that are not in x. As a consequence, all solutions of an equation have the same length.

3.2.2 A Finite-state Solver

Definition (3) yields an efficient procedure for solving analogical equations, based on finite-state transducers. The main steps of the procedure are sketched here. A full description can be found in (Yvon, 2003). To start with, let us introduce the notions of *complementary set* and *shuffle product*.

Complementary set If v is a subword of w, the *complementary set* of v with respect to w, denoted by $w\backslash v$ is the set of subwords of w obtained by removing from w, in a left-to-right fashion, the symbols in v. For example, *eea* is a complementary subword of *xmplr* with respect to *exemplar*. When v is not a subword of w, $w\backslash v$ is empty. This notion can be generalized to any regular language.

The complementary set of v with respect to w is a regular set: it is the output language of the finite-state transducer T_w (see Figure 1) for the input v.

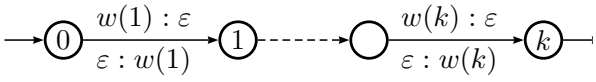


Figure 1: The transducer T_w computing complementary sets wrt w.

Shuffle The *shuffle* $u \bullet v$ of two words u and v is introduced e.g. in (Sakarovitch, 2003) as follows:

$$u \bullet v = \{u_1 v_1 u_2 v_2 \dots u_n v_n, \text{ st. } u_i, v_i \in \Sigma^\star, \\ u_1 \dots u_n = u, v_1 \dots v_n = v\}$$

The shuffle of two words u and v contains all the words w which can be composed using all the symbols in u and v, subject to the condition that if a precedes b in u (or in v), then it precedes b in w. Taking, for instance, $u = abc$ and $v = def$, the words $abcdef$, $abdefc$, $adbecf$ are in $u \bullet v$; this is not the case with $abefcd$. This operation generalizes straightforwardly to languages. The shuffle of two regular languages is regular (Sakarovitch, 2003); the automaton A, computing $K \bullet L$, is derived from the automata $A_K = (\Sigma, Q_K, q_K^0, F_K, \delta_K)$ and $A_L = (\Sigma, Q_L, q_L^0, F_L, \delta_L)$ recognizing respectively K and L as the product automata $A = (\Sigma, Q_K \times Q_L, (q_K^0, q_L^0), F_K \times F_L, \delta)$, where δ is defined as: $\delta((q_K, q_L), a) = (r_K, r_L)$ if and only if either $\delta_K(q_K, a) = r_K$ and $q_L = r_L$ or $\delta_L(q_L, a) = r_L$ and $q_K = r_K$.

The notions of complementary set and shuffle are related through the following property, which is a direct consequence of the definitions.

$$w \in u \bullet v \Leftrightarrow u \in w\backslash v$$

Solving analogical equations The notions of shuffle and complementary sets yield another characterization of analogical proportion between words, based on the following proposition:

Proposition 1.

$$\forall x, y, z, t \in \Sigma^\star, x : y :: z : t \Leftrightarrow x \bullet t \cap y \bullet z \neq \emptyset$$

An analogical proportion is thus established if the symbols in x and t are also found in y and z, and appear in the same relative order. A corollary follows:

Proposition 2.

t is a solution of $x : y :: z :?$ $\Leftrightarrow t \in (y \bullet z)\backslash x$

The set of solutions of an analogical equation $x : y :: z :?$ is a regular set, which can be computed with a finite-state transducer. It can also be shown that this analogical solver generalizes the approach based on edit distance proposed in (Lepage, 1998).

3.3 Trees

Labelled trees are very common structures in NLP tasks: they can represent syntactic structures, or terms in a logical representation of a sentence. To express the definition of analogical proportion between trees, we introduce the notion of substitution.

Definition 4 (Substitution)
A (single) substitution *is a pair* $(variable \leftarrow tree)$. *The application of the substitution* $(v \leftarrow t')$ *to a tree* t *consists in replacing each leaf of* t *labelled by* v *by the tree* t'. *The result of this operation is denoted:* $t(v \leftarrow t')$. *For each variable* v, *we define the binary operator* $\lhd_v$ *as* $t \lhd_v t' = t\,(v \leftarrow t')$.

Definition 2 can then be extended as:

Definition 5 (Analogical proportion (trees))
$(x, y, z, t) \in U$ *form an* analogical proportion, *denoted by* $x : y :: z : t$ *iff there exists some variables* $(v_1, \ldots, v_{n-1})$ *and some factorizations* $x_1 \lhd_{v_1} \ldots \lhd_{v_{n-1}} x_n = x$, $y_1 \lhd_{v_1} \ldots \lhd_{v_{n-1}} y_n = y$, $z_1 \lhd_{v_1} \ldots \lhd_{v_{n-1}} z_n = z$, $t_1 \lhd_{v_1} \ldots \lhd_{v_{n-1}} t_n = t$ *such that* $\forall i, (y_i, z_i) \in \{(x_i, t_i), (t_i, x_i)\}$.

An example of such a proportion is illustrated on Figure 2 with syntactic parse trees.

This definition yields an effective algorithm computing analogical proportions between trees (Stroppa and Yvon, 2005). We consider here a simpler heuristic approach, consisting in (i) linearizing labelled trees into parenthesized sequences of symbols and (ii) using the analogical solver for words introduced above. This approach yields a faster, albeit approximative algorithm, which makes analogical inference tractable even for large tree databases.

3.4 Algorithmic issues

We have seen how to compute analogical relationships for features whose values are words and trees.

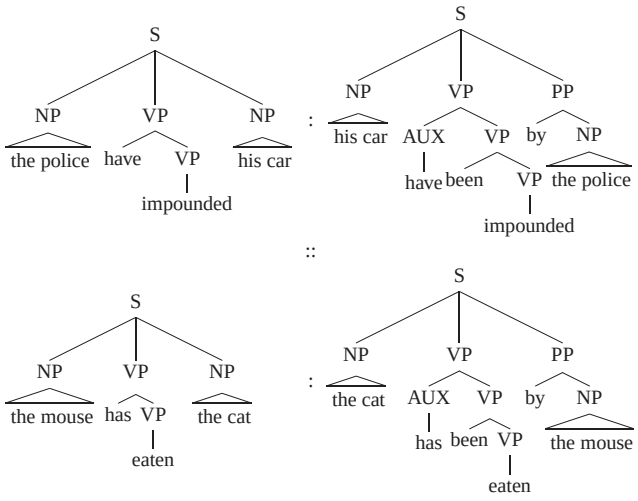


Figure 2: Analogical proportion between trees.

If we use, for trees, the solver based on tree linearizations, the resolution of an equation amounts, in both cases, to solving analogies on words.

The learning algorithm introduced in Section 2.2 is a two-step procedure: a search step and a transfer step. The latter step only involves the resolution of (a restricted number of) analogical equations. When x, y and z are known, solving $x : y :: z :?$ amounts to computing the output language of the transducer representing $(y \bullet z)\backslash x$: the automaton for this language has a number of states bounded by $|x| \times |y| \times |z|$. Given the typical length of words in our experiments, and given that the worst-case exponential bound for determinizing this automaton is hardly met, the solving procedure is quite efficient.

The problem faced during the search procedure is more challenging: given x, we need to retrieve all possible triples (y, z, t) in a finite set L such that $x : y :: z : t$. An exhaustive search requires the computation of the intersection of the finite-state automaton representing the output language of $(L \bullet L)\backslash x$ with the automaton for L. Given the size of L in our experiments (several hundreds of thousands of words), a complete search is intractable and we resort to the following heuristic approach.

L is first split into K bins $\{L_1, ..., L_K\}$, with $|L_i|$ small with respect to $|L|$. We then randomly select k bins and compute, for each bin L_i, the output language of $(L_i \bullet L_i)\backslash x$, which is then intersected with L: we thus only consider triples containing at least

two words from the same bin. It has to be noted that the bins are not randomly constructed: training examples are grouped into inflectional or derivational families. To further speed up the search, we also impose an upper bound on the degree of proportions. All triples retrieved during these k partial searches are then merged and considered for the transfer step.

The computation of analogical relationships has been implemented in a generic analogical solver; this solver is based on Vaucanson, an automata manipulation library using high performance generic programming (Lombardy et al., 2003).

4 Experiments

4.1 Methodology

The main purpose of these experiments is to demonstrate the flexibility of the analogical learner. We considered two different supervised learning tasks, both aimed at performing the lexical analysis of isolated word forms. Each of these tasks represents a possible instantiation of the learning procedure introduced in Section 2.2.

The first experiment consists in computing one or several vector(s) of morphosyntactic features to be associated with a form. Each vector comprises the lemma, the part-of-speech, and, based on the part-of-speech, additional features such as number, gender, case, tense, mood, etc. An (English) input/output pair for this tasks thus looks like: input=*replying*; output={*reply*; `V-pp--`}, where the placeholder '`-`' denotes irrelevant features. Lexical analysis is useful for many applications: a POS tagger, for instance, needs to "guess" the possible part(s)-of-speech of unknown words (Mikheev, 1997). For this task, we use the definition of analogical proportions for "flat" feature vectors (see section 3.1) and for word strings (section 3.2). The training data is a list of fully informed lexical entries; the test data is a list of isolated word forms not represented in the lexicon. Bins are constructed based on inflectional families.

The second experiment consists in computing a morphological parse of unknown lemmas: for each input lemma, the output of the system is one or several parse trees representing a possible hierarchical decomposition of the input into (morphologically categorized) morphemes (see Figure 3). This kind of analysis makes it possible to reconstruct the series of morphological operations deriving a lemma, to compute its root, its part-of-speech, and to identify morpheme boundaries. This information is required, for instance, to compute the pronunciation of an unknown word; or to infer the compositional meaning of a complex (derived or compound) lemma. Bins gather entries sharing a common root.

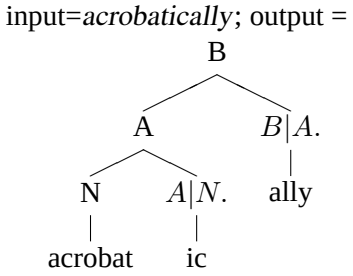


Figure 3: Input/output pair for task 2. Bound morphemes have a compositional type: $B|A.$ denotes a suffix that turns adjectives into adverbs.

These experiments use the English, German, and Dutch morphological tables of the CELEX database (Burnage, 1990). For task 1, these tables contain respectively 89 000, 342 000 and 324 000 different word forms, and the number of features to predict is respectively 6, 12, and 10. For task 2, which was only conducted with English lemma, the total number of different entries is 48 407.

For each experiment, we perform 10 runs, using 1 000 randomly selected entries for testing[1]. Generalization performance is measured as follows: the system's output is compared with the reference values (due to lexical ambiguity, a form may be associated in the database with several feature vectors or parse trees). Per instance *precision* is computed as the relative number of correct hypotheses, i.e. hypotheses which exactly match the reference: for task 1, all features have to be correct; for task 2, the parse tree has to be identical to the reference tree. Per instance *recall* is the relative number of reference values that were actually hypothesized. Precision and recall are averaged over the test set; numbers reported below are averaged over the 10 runs.

Various parameters affect the performance: k, the number of randomly selected bins considered during the search step (see Section 3.4) and d, the upper

[1]Due to lexical ambiguity, the number of tested instances is usually greater than 1 000.

bound of the degree of extracted proportions.

4.2 Experimental results

Experimental results for task 1 are given in Tables 1, 2 and 3. For each main category, two recall and precision scores are computed: one for the sole lemma and POS attributes (left column); and one for the lemma and all the morpho-syntactic features (on the right). In these experiments, parameters are set as follows: $k = 150$ and $d = 3$. As k grows, both recall and precision increase (up to a limit); $k = 150$ appears to be a reasonable trade-off between efficiency and accuracy. A further increase of d does not significantly improve accuracy: taking $d = 3$ or $d = 4$ yields very comparable results.

	Lemma + POS		Lemma + Features	
	Rec.	Prec.	Rec.	Prec.
Nouns	76.66	94.64	75.26	95.37
Verbs	94.83	97.14	94.79	97.37
Adjectives	26.68	72.24	27.89	87.67

Table 1: Results on task 1 for English

	Lemma + POS		Lemma + Features	
	Rec.	Prec.	Rec.	Prec.
Nouns	71.39	92.17	54.59	74.75
Verbs	96.75	97.85	93.26	94.36
Adjectives	91.59	96.09	90.02	95.33

Table 2: Results on task 1 for Dutch

	Lemma + POS		Lemma + Features	
	Rec.	Prec.	Rec.	Prec.
Nouns	93.51	98.28	77.32	81.70
Verbs	99.55	99.69	90.50	90.63
Adjectives	99.14	99.28	99.01	99.15

Table 3: Results on task 1 for German

As a general comment, one can note that high generalization performance is achieved for languages and categories involving rich inflectional paradigms: this is exemplified by the performance on all German categories. English adjectives, at the other end of this spectrum, are very difficult to analyze. A simple and effective workaround for this problem consists in increasing the size the sublexicons (L_i in Section 3.4) so as to incorporate in a given bin all the members of the same derivational (rather than inflectional) family. For Dutch, these results are comparable with the results reported in (van den Bosch and Daelemans, 1999), who report an accuracy of about 92% on the task of predicting the main syntactic category.

	Rec.	Prec.
Morphologically Complex	46.71	70.92
Others	17.00	46.86

Table 4: Results on task 2 for English

The second task is more challenging since the exact parse tree of a lemma must be computed. For morphologically complex lemmas (involving affixation or compounding), it is nevertheless possible to obtain acceptable results (see Table 4, showing that some derivational phenomena have been captured. Further analysis is required to assess more precisely the potential of this method.

From a theoretical perspective, it is important to realize that our model does not commit us to a morpheme-based approach of morphological processes. This is obvious in task 1; and even if task 2 aims at predicting a morphematic parse of input lemmas, this goal is achieved *without segmenting the input lemma into smaller units*. For instance, our learner parses the lemma *enigmatically* as: $[[[.N\,enigma][.A|N\,ical]]B|A.\,ly]$, that is without trying to decide to which morph the orthographic *t* should belong. In this model, input and output spaces are treated symmetrically and correspond to distinct levels of representation.

5 Discussion and future work

In this paper, we have presented a generic analogical inference procedure, which applies to a wide range of actual learning tasks, and we have detailed its instantiation for common feature types. Preliminary experiments have been conducted on two morphological analysis tasks and have shown promising generalization performance.

These results suggest that our main hypotheses are valid: (i) searching for triples is tractable even with databases containing several hundred of thousands instances; (ii) formal analogical proportions are a reliable sign of deeper analogies between lin-

guistic entities; they can thus be used to devise flexible and effective learners for NLP tasks.

This work is currently being developed in various directions: first, we are gathering additional experimental results on several NLP tasks, to get a deeper understanding of the generalization capabilities of our analogical learner. One interesting issue, not addressed in this paper, is the integration of various forms of linguistic knowledge in the definition of analogical proportions, or in the specification of the search procedure. We are also considering alternative heuristic search procedures, which could improve or complement the approaches presented in this paper. A possible extension would be to define and take advantage of non-uniform distributions of training instances, which could be used both during the searching and ranking steps. We finally believe that this approach might also prove useful in other application domains involving structured data and are willing to experiment with other kinds of data.

References

David W. Aha. 1997. Editorial. *Artificial Intelligence Review*, 11(1-5):7–10. Special Issue on Lazy Learning.

Gavin Burnage. 1990. CELEX: a guide for users. Technical report, University of Nijmegen, Center for Lexical Information, Nijmegen.

Walter Daelemans, Antal Van Den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34(1–3):11–41.

B.V. Dasarathy, editor. 1990. *Nearest neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, CA.

Brian Falkenheimer and Dedre Gentner. 1986. The structure-mapping engine. In *Proceedings of the meeting of the American Association for Artificial Intelligence (AAAI)*, pages 272–277.

Dedre Gentner, Keith J. Holyoak, and Boicho N. Konikov, editors. 2001. *The Analogical Mind*. The MIT Press, Cambridge, MA.

Douglas Hofstadter and the Fluid Analogies Research group, editors. 1995. *Fluid Concepts and Creative Analogies*. Basic Books.

Keith J. Holyoak and John E. Hummel. 2001. Understanding analogy within a biological symbol system. In Dedre Gentner, Keith J. Holyoak, and Boicho N. Konikov, editors, *The analogical mind*, pages 161–195. The MIT Press, Cambridge, MA.

Yves Lepage. 1998. Solving analogies on words: An algorithm. In *Proceedings of COLING-ACL '98*, volume 2, pages 728–735, Montréal, Canada.

Yves Lepage. 1999a. Analogy+tables=conjugation. In G. Friedl and H.G. Mayr, editors, *Proceedings of NLDB '99*, pages 197–201, Klagenfurt, Germany.

Yves Lepage. 1999b. Open set experiments with direct analysis by analogy. In *Proceedings of NLPRS '99*, volume 2, pages 363–368, Beijing, China.

Sylvain Lombardy, Raphaël Poss, Yann Régis-Gianas, and Jacques Sakarovitch. 2003. Introducing Vaucanson. In *Proceedings of CIAA 2003*, pages 96–107.

Andrei Mikheev. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405–423.

Vito Pirrelli and François Yvon. 1999. Analogy in the lexicon: a probe into analogy-based machine learning of language. In *Proceedings of the 6th International Symposium on Human Communication*, Santiago de Cuba, Cuba.

Tony A. Plate. 2000. Analogy retrieval and processing with distributed vector representations. *Expert systems*, 17(1):29–40.

Jacques Sakarovitch. 2003. *Eléments de théorie des automates*. Vuibert, Paris.

Royal Skousen. 1989. *Analogical Modeling of Language*. Kluwer, Dordrecht.

Nicolas Stroppa and François Yvon. 2005. Formal models of analogical relationships. Technical report, ENST, Paris, France.

Paul Thagard, Keith J. Holoyak, Greg Nelson, and David Gochfeld. 1990. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46(3):259–310.

Antal van den Bosch and Walter Daelemans. 1999. Memory-based morphological processing. In *Proceedings of ACL*, pages 285–292, Maryland.

D. Randall Wilson and Tony R. Martinez. 2000. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38(3):257–286.

François Yvon. 1999. Pronouncing unknown words using multi-dimensional analogies. In *Proc. Eurospeech*, volume 1, pages 199–202, Budapest, Hungary.

François Yvon. 2003. Finite-state machines solving analogies on words. Technical report, ENST.

Morphology Induction From Term Clusters

Dayne Freitag
HNC Software, LLC
3661 Valley Centre Drive
San Diego, CA 92130, USA
daynefreitag@fairisaac.com

Abstract

We address the problem of learning a morphological automaton directly from a monolingual text corpus without recourse to additional resources. Like previous work in this area, our approach exploits orthographic regularities in a search for possible morphological segmentation points. Instead of affixes, however, we search for affix transformation rules that express correspondences between term clusters induced from the data. This focuses the system on substrings having syntactic function, and yields cluster-to-cluster transformation rules which enable the system to process unknown morphological forms of known words accurately. A stem-weighting algorithm based on Hubs and Authorities is used to clarify ambiguous segmentation points. We evaluate our approach using the CELEX database.

1 Introduction

This paper presents a completely unsupervised method for inducing morphological knowledge directly from a large monolingual text corpus. This method works by searching for transformation rules that express correspondences between term clusters which are induced from the corpus in an initial step. It covers both inflectional and derivational morphology, and is able to process previously unseen morphs of a word, as long as one of its morphs has been assigned to a cluster.

Aside from its academic appeal, acquisition of this morphological knowledge is a step toward the goal of rapidly retargetable natural language processing. Toward this end, we envisage two uses for it:

1. It can be used to perform morphological normalization (i.e., *stemming* (Porter, 1980)).

2. In the form of *transformation rules*, it can help us classify unknown words, thereby enhancing the utility of cluster-based features for applications such as information extraction (Miller et al., 2004; Freitag, 2004).

There is a considerable literature on the problem of morphology induction in general, and unsupervised (or lightly supervised) induction in particular. Much of the work attempts to exploit orthographic regularities alone, seeking affixation patterns (or *signatures*) that permit a compressive representation of the corpus. Several researchers propose algorithms based on the minimum description length (MDL) principle, achieving reasonable success in discovering regular morphological patterns (Brent et al., 1995; Goldsmith, 2000; Creutz and Lagus, 2002; Argamon et al., 2004). MDL has information theoretic underpinnings, and an information theoretic objective function achieves similar success (Snover et al., 2002). Note that none of these approaches attempts to account for the syntactic dimension of affixation. And all must adopt strategies to cope with a very large search space (the power set of the vocab-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 128–135, Ann Arbor, June 2005.

ulary, in the limit). Such strategies form a common theme in these papers.

Our approach implicitly employs term co-occurrence statistics in the form of statistically derived term clusters. A number of researchers use such statistics directly. A common technique is to cast a word as a distribution over other words that occur within some limited window across the corpus. This definition of co-occurrence yields a semantic distance measure which tends to draw together inflectional variants of a word. Combined with heuristics such as string edit distance, it can be used to find reliable conflation sets (Xu and Croft, 1998; Baroni et al., 2002). A somewhat tighter definition of co-occurrence, which nevertheless yields a semantic distance measure, serves as the basis of a method that captures *irregular* inflectional transformations in Yarowsky and Wicentowski (2001).[1] Schone and Jurafsky (2001) employ distributions over adjacent words (yielding a syntactic distance metric) to improve the precision of their conflation sets.

In contrast with these approaches, ours is predicated on a strictly local notion of co-occurrence. It is well known that clustering terms from a corpus in English or a related language, using a distance measure based on local co-occurrence, yields clusters that correspond roughly to part of speech categories (Schütze, 1995; Clark, 2000). The heart of our idea is to search for affix transformation rules mapping terms in one cluster to those in another. The search for such rules has previously been conducted in the context of supervised part-of-speech tagging (Mikheev, 1997), but not to our knowledge using word clusters. Basing the search for affix patterns on a syntactic partition of the vocabulary, albeit a noisy one, greatly reduces the size of the space of possible conflation sets. Furthermore, the resulting rules can be assigned a syntactic interpretation.

2 Clustering

A prerequisite of our method is a clustering of terms in the corpus vocabulary into rough syntactic groups. To achieve this, we first collect co-occurrence statistics for each word, measuring the frequency of words found immediately adjacent to it in the corpus, treating left occurrences as distinct from right occurrences. This co-occurrence database serves as input to *information theoretic co-clustering* (Dhillon et al., 2003), which seeks a partition of the vocabulary that maximizes the mutual information between term categories and their contexts. This approach to term clustering is closely related to others from the literature (Brown et al., 1992; Clark, 2000).[2]

`recently soon slightly quickly` …
`underwriter designer commissioner` …
`increased posted estimated raised` …
`agreed declined expects wants` …

Table 1: Sample members of four clusters from the Wall Street Journal corpus.

Recall that the *mutual information* between random variables X and Y can be written:

$$\mathrm{M}_{XY} = \sum_{xy} \mathrm{P}(x,y) \ln \frac{\mathrm{P}(x,y)}{\mathrm{P}(x)\mathrm{P}(y)} \qquad (1)$$

Here, X and Y correspond to term and context clusters, respectively, each event x and y the observation of some term and contextual term in the corpus. We perform an approximate maximization of M_{XY} using a simulated annealing procedure in which each random trial move takes a word x or context y out of the cluster to which it is tentatively assigned and places it into another.

We performed this procedure on the Wall Street Journal (WSJ) portion of the North American News corpus, forming 200 clusters. Table 1 shows sample terms from several hand-selected clusters.

3 Method

In our experiments and the discussion that follows, *stems* are sub-strings of words, to which attach *affixes*, which are sub-string classes denoted by perl-style regular expressions (e.g., `e?d$` or `^re`). A *transform* is an affix substitution which entails a change of clusters. We depict the affix part of the

[1] Note that this method presupposes the availability of several resources in addition to a corpus, including a list of canonical inflectional suffixes.

[2] While we have not experimented with other clustering approaches, we assume that the accuracy of the derived morphological information is *not* very sensitive to the particular methodology.

transform using a perl-style `s///` operator. For example, the transform `s/ed$/ing/` corresponds to the operation of replacing the suffix `ed` with `ing`.

3.1 Overview

The process of moving from term clusters to a transform automaton capable of analyzing novel forms consists of four stages:

1. **Acquire candidate transformations**. By searching for transforms that align a large number of terms in a given pair of clusters, we quickly identify affixation patterns that are likely to have syntactic significance.

2. **Weighting stems and transforms**. The output of Step 1 is a set of transforms, some overlapping, others dubious. This step weights them according to their utility across the vocabulary, using an algorithm similar to Hubs and Authorities (Kleinberg, 1998).

3. **Culling transforms**. We segment the words in the vocabulary, using the transform weights to choose among alternative segmentations. Following this segmentation step, we discard any transform that failed to participate in at least one segmentation.

4. **Constructing an automaton**. From the remaining transforms we construct an automaton, the nodes of which correspond to clusters, the edges to transforms. The resulting data structure can be used to construct morphological parses.

The remainder of this section describes each of these steps in detail.

3.2 Acquiring Transforms

Once we are in possession of a sufficiently large number of term clusters, the acquisition of candidate transforms is conceptually simple. For each pair of clusters, we count the number of times each possible transform is in evidence, then discard those transforms occurring fewer than some small number of times.

For each pair of clusters, we search for suffix or prefix pairs, which, when stripped from matching members in the respective clusters lead to as large a cluster intersection as possible. For example, if `walked` and `talked` are in Cluster 1, and `walking` and `talking` are in Cluster 2, then `walk` and `talk` are in the intersection, given the transform `s/ed$/ing/`. In our experiments, we retain any cluster-to-cluster transform producing an intersection having at least three members.

`s/ful$/less/`	`pain harm use` ...
`s/^/over/`	`charged paid hauled` ...
`s/cked$/wing/`	`kno sho che` ...
`s/nd$/ts/`	`le se fi` ...
`s/s$/ed/`	`recall assert add` ...
`s/ts$/ted/`	`asser insis predic` ...
`s/es$/ing/`	`argu declar acknowledg` ...
`s/s$/ing/`	`recall assert add` ...

Table 2: Sample transforms and matching stems from the Wall Street Journal after the acquisition step.

Table 2 lists some transforms derived from the WSJ as part of this process, along with a few of the stems they match. These were chosen for the sake of illustration; this list does not necessarily the reflect the quality or distribution of the output. (For example, transforms based on the pattern `s/$/s/` easily form the largest block.)

A frequent problem is illustrated by the transforms `s/s$/ed/` and `s/ts$/ted/`. Often, we observe alternative segmentations for the same words and must decide which to prefer. We resolve most of these questions using a simple heuristic. If one transform subsumes another—if the vocabulary terms it covers is a strict superset of those covered by the other transform—then we discard the second one. In the table, all members of the transform `s/ts$/ted/` are also members of `s/s$/ed/`, so we drop `s/ts$/ted/` from the set.

The last two lines of the table represent an obvious opportunity to generalize. In cases like this, where two transforms are from the same cluster pair and involve source or destination affixes that differ in a single letter, the other affixes being equal, we introduce a new transform in which the elided letter is optional (in this example, the transform `s/e?s$/ing/`). The next step seeks to resolve this uncertainty.

s/$/s/	0.2
s/e?$/ed/	0.1
s/e?$/ing/	0.1
s/s$/ses/	1.6e-14
s/w$/ws/	1.6e-14
s/^b/c/	1.6e-14

Table 3: The three highest-weighted and lowest-weighted transforms.

3.3 Weighting Stems and Transforms

The observation that morphologically significant affixes are more likely to be frequent than arbitrary word endings is central to MDL-based systems. Of course, the same can be said about word stems: a string is more likely to be a stem if it is observed with a variety of affixes (or transforms). Moreover, our certainty that it is a valid stem increases with our confidence that the affixes we find attached to it are valid.

This suggests that candidate affixes and stems can "nominate" each other in a way analogous to "hubs" and "authorities" on the Web (Kleinberg, 1998). In this step, we exploit this insight in order to weight the "stem-ness" and "affix-ness" of candidate strings. Our algorithm is closely based on the Hubs and Authorities Algorithm. We say that a stem and transform are "linked" if we have observed a stem to participate in a transform. Beginning with a uniform distribution over stems, we zero the weights associated with transforms, then propagate the stem weights to the transforms. For each stem S and transform T, such that S and T are linked, the weight of S is added to the weight of T. Next, the stem weights are zeroed, and the transform weights propagated to the stems in the same way. This procedure is iterated a few times or until convergence (five times in these experiments).

3.4 Culling Transforms

The output of this procedure is a weighting of candidate stems, on the one hand, and transforms, on the other. Table 3 shows the three highest-weighted and three lowest-weighted transforms from an experiment involving the 10,000 most frequent words in the WSJ.

Although these weights have no obvious linguistic interpretation, we nevertheless can use them to filter further the transform set. In general, however, there is no single threshold that removes all dubious transforms. It does appear to hold, though, that correct transforms (e.g., s/$/s/) outweigh competing incorrect transforms (e.g., s/w$/ws/). This observation motivates our culling procedure: We apply the transforms to the vocabulary in a competitive segmentation procedure, allowing highly weighted transforms to "out-vote" alternative transforms with lower weights. At the completion of this pass through the vocabulary, we retain only those transforms that contribute to at least one successful segmentation.

```
1:  procedure SEGMENT(w)
2:      A[*] ← ∅          ▷ Expansions to transform sets
3:      S[*] ← 0                    ▷ Stems to scores
4:      for each transform t do
5:          if there exists s s.t. w ∈ s · t then
6:              A[s · t] ← A[s · t] ∪ {t}
7:          end if
8:      end for
9:      for T ∈ Range(A) do
10:         B[*] ← 0
11:         for t ∈ T do
12:             s ← w ÷ t
13:             B[s] ← B[s] + Weight(t)
14:         end for
15:         s ← MaxArg_x B[x]
16:         S[s] ← S[s] + B[s]
17:     end for
18:     return MaxArg_x S[x]
19: end procedure
```

Table 4: The segmentation procedure.

Table 4 lists the segmentation procedure. In this pseudocode, w is a word, t a transform, and s a stem. The operation $s \cdot t$ produces the set of (two) words generated by applying the affixes of t to s; the operation $w \div t$ (the *stemming* operation) removes the longest matching affix of t from w. Given a word w, we first find the set of transforms associated with w, grouping them by the pair of words to which they correspond (Lines 4–8). For example, given the word "created", and the transforms s/ed$/ing/, s/ted$/ting/, and s/s$/d/,

the first two transforms will be grouped together in A (with index $\{\text{created}, \text{creating}\}$), while the third will be part of a different group.

Once we have grouped associated transforms, we use them to stem w, accumulating evidence for different stemmings in B. In Line 15, we then discard all but the highest scoring stemming. The score of this stemming is then added to its "global" score in Line 16.

The purpose of this procedure is the suppression of spurious segmentations in Line 15. Although this pseudocode returns only the highest weighted segmentation, it is usually the case that *all* candidate segmentations stored in S are valid, i.e., that several or all breakpoints of a product of multiple affixation are correctly found. And it is a byproduct of this procedure that we require for the final step in our pipeline: In addition to accumulating stemming scores, we record the transforms that contributed to them. We refer to this set of transforms as the *culled* set.

3.5 Constructing an Automaton

Given the culled set of transforms, creation of a parser is straightforward. In the last two steps we have considered a transform to be a pair of affixes (A_S, A_D). Recall that for each such transform there are one or more cluster-specific transforms of the form (C_S, A_S, C_D, A_D) in which the source and destination affixes correspond to clusters. We now convert this set of specific transforms into an automaton in which clusters form the nodes and arcs are affixation operations. For every transform (C_S, A_S, C_D, A_D), we draw an arc from C_S to C_D, labeling it with the general transform (A_S, A_D), and draw the inverse arc from C_D to C_S.

We can now use this automaton for a kind of unsupervised morphological analysis. Given a word, we construct an analysis by finding paths through the automaton to known (or possibly unknown) stem words. Each step replaces one (possibly empty) affix with another one, resulting in a new word form. In general, many such paths are possible. Most of these are redundant, generated by following given affixation arcs to alternative clusters (there are typically several plural noun clusters, for example) or collapsing compound affixations into a single operation.

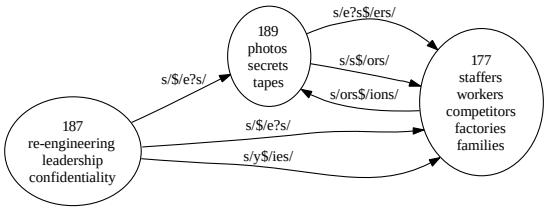


Figure 1: A fragment of the larger automaton from the Wall Street Journal corpus.

In our experiments, we generate all possible paths under the constraint that an operation lead to a known longer wordform, that it be a possible stem of the given word, and that the operation not constitute a loop in the search.[3] We then sort the analysis traces heuristically and return the top one as our analysis. In comparing two traces, we use the following criteria, in order:

- Prefer the trace with the shorter starting stem.
- Prefer the trace involving fewer character edits. (The number of edits is summed across the traces, the trace with the smaller sum preferred.)
- Prefer the trace having more correct cluster assignments of intermediate wordforms.
- Prefer the longer trace.

Note that it is not always clear how to perform an affixation. Consider the transform `s/ing$/e?d/`, for example. In practice, however, this is not a source of difficulty. We attempt both possible expansions (with or without the "e"). If either produces a known wordform which is found in the destination cluster, we discard the other one. If neither resulting wordform can be found in the destination cluster, both are added to the frontier in our search.

4 Evaluation

We evaluate by taking the highest-ranked trace, using the ordering heuristics described in the previous section, as the system's analysis of a given

[3] One wordform w_1 is a possible stem of another w_2, if after stripping any of the affixes in the culled set the resulting string is a sub-string of w_2.

word. This analysis takes the form of a sequence of hypothetical wordforms, from a putative stem to the target wordform (e.g., `decide`, `decision`, `decisions`). The CELEX morphological database (Baayen et al., 1995) is used to produce a reference analysis, by tracing back from the target wordform through any inflectional affixation, then through successive derivational affixations until a stem is reached. Occasionally, this yields more than one analysis. In such cases, all analyses are retained, and the system's analysis is given the most optimistic score. In other words, if a CELEX analysis is found which matches the system's analysis, it is judged to be correct.

4.1 Results

In evaluating an analysis, we distinguish the following outcomes (ordered from most favorable to least):

- **Cor**. The system's analysis matches CELEX's.
- **Over**. The system's analysis contains all the wordforms in CELEX's, also contains additional wordforms, and each of the wordforms is a legitimate morph of the CELEX stem.
- **Under**. The system's analysis contains some of the wordforms in CELEX's; it may contain additional wordforms which are legitimate morphs of the CELEX stem. This happens, for example, when the CELEX stem is unknown to the system.
- **Fail**. The system failed to produce an analysis for a word for which CELEX produced a multiwordform analysis.
- **Spur**. The system produced an analysis for a word which CELEX considered a stem.
- **Incor**. All other (incorrect) cases.

Note that we discard any wordforms which are not in CELEX. Depending on the vocabulary size, anywhere from 15% to 30% are missing. These are often proper nouns.

In addition, we measure precision, recall, and F1 as in Schone and Jurafsky (2001). These metrics reflect the algorithm's ability to group known terms which are morphologically related. Groups are formed by collecting all wordforms that, when analyzed, share a root form. We report these numbers as **Prec**, **Rec**, and **F1**.

	1K	5K	10K	10K+1K	20K
Cor	0.74	0.74	0.75	0.64	0.71
Over	0	0.004	0.003	0.002	0.002
Under	0.005	0.04	0.05	0.06	0.07
Fail	0.25	0.21	0.18	0.28	0.14
Spur	0	0.002	0.01	0.01	0.02
Incor	0	0.003	0.01	0.02	0.05
Prec	1.0	0.98	0.95	1.0	0.80
Rec	0.85	0.82	0.81	0.96	0.82
F1	0.92	0.90	0.87	0.98	0.81

Table 5: Results of experiments using the Wall Street Journal corpus.

We performed the procedure outlined in Section 3.1 using the k most frequent terms from the Wall Street Journal corpus, for k ranging from 1000 to 20,000. The expense of performing these steps is modest compared with that of collecting term co-occurrence statistics and generating term clusters. Our perl implementation of this procedure consumes just over two minutes on a lightly loaded 2.5 GHz Intel machine running Linux, given a collection of 10,000 wordforms in 200 clusters.

The header of each column in Table 5 displays the size of the vocabulary. The column labeled *10K+1K* stands for an experiment designed to assess the ability of the algorithm to process novel terms. For this column, we derived the morphological automaton from the 10,000 most frequent terms, then used it to analyze the next 1000 terms.

The surprising precision/recall scores in this column—scores that are high despite an actual degradation in performance—argues for caution in the use and interpretation of the precision/recall metrics in this context. The difficulty of the morphological conflation set task is a function of the size and constituency of a vocabulary. With a small sample of terms relatively low on the Zipf curve, high precision/recall scores mainly reflect the algorithm's ability to determine that most of the terms are *not* related—a Pyrrhic victory. Nevertheless, these metrics give us a point of comparison with Schone and Jurafsky (2001) who, using a vocabulary of English words occurring at least 10 times in a 6.7 million-word newswire corpus, report F1 of 88.1 for con-

flation sets based only on suffixation, and 84.5 for circumfixation. While a direct comparison would be dubious, the results in Table 5 are comparable to those of Schone and Jurafsky. (Note that we include both prefixation and suffixation in our algorithm and evaluation.)

Not surprisingly, precision and recall degrade as the vocabulary size increases. The top rows of the table, however, suggest that performance is reasonable at small vocabulary sizes and robust across the columns, up to 20K, at which point the system increasingly generates incorrect analyses (more on this below).

4.2 Discussion

A primary advantage of basing the search for affixation patterns on term clusters is that the problem of non-morphological orthographic regularities is greatly mitigated. Nevertheless, as the vocabulary grows, the inadequacy of the simple frequency thresholds we employ becomes clear. In this section, we speculate briefly about how this difficulty might be overcome.

At the 20K size, the system identifies and retains a number of non-morphological regularities. An example are the transforms `s/$/e/` and `s/$/o/`, both of which align members of a name cluster with other members of the same cluster (Clark/Clarke, Brook/Brooke, Robert/Roberto, etc.). As a consequence, the system assigns the analysis `tim => time` to the word "time", suggesting that it be placed in the name cluster.

There are two ways in which we can attempt to suppress such analyses. One is to adjust parameters so that noise transforms are less likely. The procedure for acquiring candidate transforms, described in Section 3.2, discards any that match fewer than 3 stems. When we increase this parameter to 5 and run the 20K experiment again, the incorrect rate falls to 0.02 and F1 rises to 0.84. While this does not solve the larger problem of spurious transforms, it does indicate that a search for a more principled way to screen transforms should enhance performance.

The other way to improve analyses is to corroborate predictions they make about the constituent wordforms. If the `tim => time` analysis is correct, then the word "time" should be at home in the name cluster. This is something we can check. Recall that in our framework both terms and clusters are associated with distributions over adjacent terms (or clusters). We can hope to improve precision by discarding analyses that assign a term to a cluster from which it is too distributionally distant. Applying such a filter in the 20K experiment, has a similar impact on performance as the transform filter of the previous paragraph, with F1 rising to 0.84.[4]

Several researchers have established the utility of a filter in which the broader context distributions surrounding two terms are compared, in an effort to insure that they are semantically compatible (Schone and Jurafsky, 2001; Yarowsky and Wicentowski, 2001). This would constitute a straightforward extension of our framework.

Note that the system is often able to produce the correct analysis, but ordering heuristics described in Section 3.5 cause it to be discarded in favor of an incorrect one. The analyses `us => using` and `use => using` are an example, the former being the one favored for the word "using". Note, though, that our automaton construction procedure discards a potentially useful piece of information—the amount of support each arc receives from the data (the number of stems it matches). This might be converted into something like a traversal probability and used in ordering analyses.

Of course, a further shortcoming of our approach is its inability to account for irregular forms. It shares this limitation with all other approaches based on orthographic similarity (a notable exception is Yarowsky and Wicentowski (2001)). However, there is reason to believe that it could be extended to accommodate at least some irregular forms. We note, for example, the cluster pair 180/185, which is dominated by the transform `s/e?$/ed/`. Cluster 180 contains words like "make", "pay", and "keep", while Cluster 185 contains "made", "paid", and "kept". In other words, once a strong correspondence is found between two clusters, we can search for an alignment which covers the orphans in the respective clusters.

[4] Specifically, we take the Hellinger distance between the two distributions, scaled into the range $[0, 1]$, and discard those analyses for which the term is at a distance greater than 0.5 from the proposed cluster.

5 Conclusion

We have shown that automatically computed term clusters can form the basis of an effective unsupervised morphology induction system. Such clusters tend to group terms by part of speech, greatly simplifying the search for syntactically significant affixes. Furthermore, the learned affixation patterns are not just orthographic features or morphological conflation sets, but cluster-to-cluster transformation rules. We exploit this in the construction of morphological automata able to analyze previously unseen wordforms.

We have not exhausted the sources of evidence implicit in this framework, and we expect that attending to features such as transform frequency will lead to further improvements. Our approach may also benefit from the kinds of broad-context semantic filters proposed elsewhere. Finally, we hope to use the cluster assignments suggested by the morphological rules in refining the original cluster assignments, particularly of low-frequency words.

Acknowledgments

This material is based on work funded in whole or in part by the U.S. Government. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the U.S. Government.

References

S. Argamon, N. Akiva, A. Amir, and O. Kapah. 2004. Efficient unsupervised recursive word segmentation using minimum description length. In *Proc. 20th International Conference on Computational Linguistics (Coling-04)*.

R.H. Baayen, R. Piepenbrock, and L. Gulikers. 1995. *The CELEX Lexical Database (CD-ROM)*. LDC, University of Pennsylvania, Philadelphia.

M. Baroni, J. Matiasek, and H. Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proc. ACL-02 Workshop on Morphological and Phonological Learning*.

M. Brent, S.K. Murthy, and A. Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proc. 5th International Workshop on Artificial Intelligence and Statistics*.

P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, and R.L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *CoNLL 2000*, September.

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Morphological and Phonological Learning: Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*.

I.S. Dhillon, S. Mallela, and D.S. Modha. 2003. Information-theoretic co-clustering. Technical Report TR-03-12, Dept. of Computer Science, U. Texas at Austin.

D. Freitag. 2004. Trained named entity recognition using distributional clusters. In *Proceedings of EMNLP-04*.

J. Goldsmith. 2000. Unsupervised learning of the morphology of a natural language. http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000/Paper/paper.html.

J.M. Kleinberg. 1998. Authoritative sources in a hyperlinked environment. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*.

A. Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT/NAACL 04*.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

P. Schone and D. Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proc. NAACL-01*.

H. Schütze. 1995. Distributional part-of-speech tagging. In *Proc. 7th EACL Conference (EACL-95)*, March.

M.G. Snover, G.E. Jarosz, and M.R. Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In *Morphological and Phonological Learning: Proc. 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*.

J. Xu and W.B. Croft. 1998. Corpus-based stemming using co-occurrence of word variants. *ACM TOIS*, 18(1).

D. Yarowsky and R. Wicentowski. 2001. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL-01*.

Beyond the Pipeline: Discrete Optimization in NLP

Tomasz Marciniak and Michael Strube
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp

Abstract

We present a discrete optimization model based on a linear programming formulation as an alternative to the *cascade of classifiers* implemented in many language processing systems. Since NLP tasks are correlated with one another, sequential processing does not guarantee optimal solutions. We apply our model in an NLG application and show that it performs better than a pipeline-based system.

1 Introduction

NLP applications involve mappings between complex representations. In *generation* a representation of the semantic content is mapped onto the grammatical form of an expression, and in *analysis* the semantic representation is derived from the linear structure of a text or utterance. Each such mapping is typically split into a number of different tasks handled by separate modules. As noted by Daelemans & van den Bosch (1998), individual decisions that these tasks involve can be formulated as classification problems falling in either of two groups: *disambiguation* or *segmentation*. The use of machine-learning to solve such tasks facilitates building complex applications out of many *light* components. The architecture of choice for such systems has become a *pipeline*, with strict ordering of the processing stages. An example of a generic pipeline architecture is GATE (Cunningham et al., 1997) which provides an infrastructure for building NLP applications. Sequential processing has also been used in several NLG systems (e.g. Reiter (1994), Reiter & Dale (2000)), and has been successfully used to combine standard preprocessing tasks such as part-of-speech tagging, chunking and named entity recognition (e.g. Buchholz et al. (1999), Soon et al. (2001)).

In this paper we address the problem of aggregating the outputs of classifiers solving different NLP tasks. We compare pipeline-based processing with discrete optimization modeling used in the field of computer vision and image recognition (Kleinberg & Tardos, 2000; Chekuri et al., 2001) and recently applied in NLP by Roth & Yih (2004), Punyakanok et al. (2004) and Althaus et al. (2004). Whereas Roth and Yih used optimization to solve two tasks only, and Punyakanok et al. and Althaus et al. focused on a single task, we propose a general formulation capable of combining a large number of different NLP tasks. We apply the proposed model to solving numerous tasks in the generation process and compare it with two pipeline-based systems.

The paper is structured as follows: in Section 2 we discuss the use of classifiers for handling NLP tasks and point to the limitations of pipeline processing. In Section 3 we present a general discrete optimization model whose application in NLG is described in Section 4. Finally, in Section 5 we report on the experiments and evaluation of our approach.

2 Solving NLP Tasks with Classifiers

Classification can be defined as the task T_i of assigning one of a discrete set of m_i possible labels $L_i = \{l_{i1}, ..., l_{im_i}\}$[1] to an unknown instance. Since generic machine-learning algorithms can be applied to solving single-valued predictions only, complex

[1]Since we consider different NLP tasks with varying numbers of labels we denote the cardinality of L_i, i.e. the set of possible labels for task T_i, as m_i.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 136–143, Ann Arbor, June 2005.

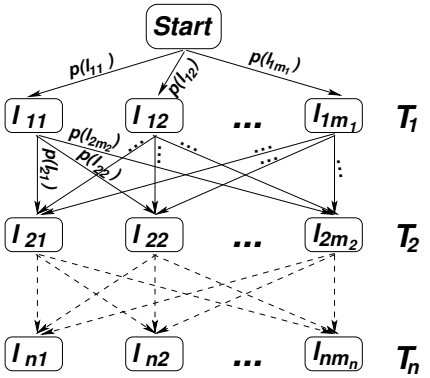


Figure 1: Sequential processing as a graph.

structures, such as parse trees, coreference chains or sentence plans, can only be assembled from the outputs of many different classifiers.

In an application implemented as a *cascade of classifiers* the output representation is built incrementally, with subsequent classifiers having access to the outputs of previous modules. An important characteristic of this model is its extensibility: it is generally easy to change the ordering or insert new modules at any place in the pipeline[2]. A major problem with sequential processing of linguistic data stems from the fact that elements of linguistic structure, at the semantic or syntactic levels, are strongly correlated with one another. Hence classifiers that have access to additional contextual information perform better than if this information is withheld. In most cases, though, if task T_k can use the output of T_i to increase its accuracy, the reverse is also true. In practice this type of processing may lead to *error propagation*. If due to the scarcity of contextual information the accuracy of initial classifiers is low, erroneous values passed as input to subsequent tasks can cause further misclassifications which can distort the final outcome (also discussed by Roth and Yih and van den Bosch et al. (1998)).

As can be seen in Figure 1, solving classification tasks sequentially corresponds to the *best-first* traversal of a weighted multi-layered lattice. Nodes at separate layers $(T_1, ..., T_n)$ represent labels of different classification tasks and transitions between the nodes are augmented with probabilities of selecting respective labels at the next layer. In the sequential model only transitions between nodes belonging to subsequent layers are allowed. At each step, the transition with the highest *local* probability is selected. Selected nodes correspond to outcomes of individual classifiers. This graphical representation shows that sequential processing does not guarantee an optimal context-dependent assignment of class labels and favors tasks that occur later, by providing them with contextual information, over those that are solved first.

3 Discrete Optimization Model

As an alternative to sequential ordering of NLP tasks we consider the *metric labeling problem* formulated by Kleinberg & Tardos (2000), and originally applied in an image restoration application, where classifiers determine the "true" intensity values of individual pixels. This task is formulated as a labeling function $f : P \rightarrow L$, that maps a set P of n objects onto a set L of m possible labels. The goal is to find an assignment that minimizes the overall cost function $Q(f)$, that has two components: *assignment costs*, i.e. the costs of selecting a particular label for individual objects, and *separation costs*, i.e. the costs of selecting a pair of labels for two *related* objects[3]. Chekuri et al. (2001) proposed an integer linear programming (ILP) formulation of the metric labeling problem, with both assignment cost and separation costs being modeled as binary variables of the linear cost function.

Recently, Roth & Yih (2004) applied an ILP model to the task of the simultaneous assignment of semantic roles to the entities mentioned in a sentence and recognition of the relations holding between them. The assignment costs were calculated on the basis of predictions of *basic* classifiers, i.e. trained for both tasks individually with no access to the outcomes of the other task. The separation costs were formulated in terms of binary constraints, that specified whether a specific semantic role could occur in a given relation, or not.

In the remainder of this paper, we present a more general model, that is arguably better suited to handling different NLP problems. More specifically, we

[2]Both operations only require retraining classifiers with a new selection of the input features.

[3]These costs were calculated as the function of the metric distance between a pair of pixels and the difference in intensity.

put no limits on the number of tasks being solved, and express the separation costs as stochastic constraints, which for almost any NLP task can be calculated off-line from the available linguistic data.

3.1 ILP Formulation

We consider a general context in which a specific NLP problem consists of individual linguistic decisions modeled as a set of n classification tasks $T = \{T_1, ..., T_n\}$, that potentially form mutually related pairs. Each task T_i consists in assigning a label from $L_i = \{l_{i1}, ..., l_{im_i}\}$ to an instance that represents the particular decision. Assignments are modeled as variables of a linear cost function. We differentiate between *simple variables* that model individual assignments of labels and *compound variables* that represent respective assignments for each pair of related tasks.

To represent individual assignments the following procedure is applied: for each task T_i, every label from L_i is associated with a binary variable $x(l_{ij})$. Each such variable represents a binary choice, i.e. a respective label l_{ij} is selected if $x(l_{ij}) = 1$ or rejected otherwise. The coefficient of variable $x(l_{ij})$, that models the assignment cost $c(l_{ij})$, is given by:

$$c(l_{ij}) = -log_2(p(l_{ij}))$$

where $p(l_{ij})$ is the probability of l_{ij} being selected as the outcome of task T_i. The probability distribution for each task is provided by the basic classifiers that do not consider the outcomes of other tasks[4].

The role of compound variables is to provide pairwise constraints on the outcomes of individual tasks. Since we are interested in constraining only those tasks that are truly dependent on one another we first apply the contingency coefficient C to measure the degree of correlation for each pair of tasks[5].

In the case of tasks T_i and T_k which are *significantly correlated*, for each pair of labels from $L_i \times L_k$ we build a single variable $x(l_{ij}, l_{kp})$. Each such variable is associated with a coefficient representing the constraint on the respective pair of labels l_{ij}, l_{kp} calculated in the following way:

$$c(l_{ij}, l_{kp}) = -log_2(p(l_{ij,}l_{kp}))$$

with $p(l_{ij,}l_{kp})$ denoting the *prior* joint probability of labels $l_{ij,}$ and l_{kp} in the data, which is independent from the general classification context and hence can be calculated off-line[6].

The ILP model consists of the target function and a set of constraints which block *illegal* assignments (e.g. only one label of the given task can be selected)[7]. In our case the target function is the cost function $Q(f)$, which we want to minimize:

$$min \;\; Q(f) = \sum_{T_i \in T} \sum_{l_{ij} \in L_i} c(l_{ij}) \cdot x(l_{ij})$$

$$+ \sum_{T_i, T_k \in T, i<k} \sum_{l_{ij}, l_{kp} \in L_i \times L_k} c(l_{ij}, l_{kp}) \cdot x(l_{ij}, l_{kp})$$

Constraints need to be formulated for both the simple and compound variables. First we want to ensure that exactly one label l_{ij} belonging to task T_i is selected, i.e. only one simple variable $x(l_{ij})$ representing labels of a given task can be set to 1:

$$\sum_{l_{ij} \in L_i} x(l_{ij}) = 1, \qquad \forall i \in \{1, ..., n\}$$

We also require that if two simple variables $x(l_{ij})$ and $x(l_{kp})$, modeling respectively labels l_{ij} and l_{kp} are set to 1, then the compound variable $x(l_{ij}, l_{kp})$, which models co-occurrence of these labels, is also set to 1. This is done in two steps: we first ensure that if $x(l_{ij}) = 1$, then exactly one variable $x(l_{ij}, l_{kp})$ must also be set to 1:

$$x(l_{ij}) - \sum_{l_{kp} \in L_k} x(l_{ij}, l_{kp}) = 0,$$

$$\forall i, k \in \{1, ..., n\}, i < k \;\wedge\; j \in \{1, ..., m_i\}$$

and do the same for variable $x(l_{kp})$:

[4] In this case the ordering of tasks is not necessary, and the classifiers can run independently from each other.

[5] C is a test for measuring the association of two nominal variables, and hence adequate for the type of tasks that we consider here. The coefficient takes values from 0 (no correlation) to 1 (complete correlation) and is calculated by the formula: $C = (\chi^2/(N + \chi^2))^{1/2}$, where χ^2 is the chi-squared statistic and N the total number of instances. The significance of C is then determined from the value of χ^2 for the given data. See e.g. Goodman & Kruskal (1972).

[6] In Section 5 we discuss an alternative approach which considers the actual input.

[7] For a detailed overview of linear programming and different types of LP problems see e.g. Nemhauser & Wolsey (1999).

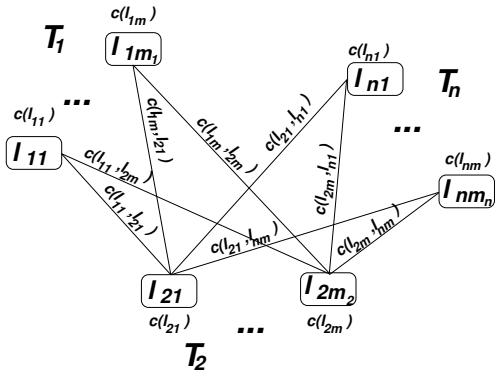


Figure 2: Graph representation of the ILP model.

$$x(l_{kp}) - \sum_{l_{ij} \in L_i} x(l_{ij}, l_{kp}) = 0,$$

$$\forall i, k \in \{1, ..., n\}, i < k \ \wedge \ p \in \{1, ..., m_k\}$$

Finally, we constrain the values of both simple and compound variables to be binary:

$$x(l_{ij}) \in \{0, 1\} \ \wedge \ x(l_{ij}, l_{kp}) \in \{0, 1\},$$

$$\forall i, k \in \{1, ..., n\} \ \wedge \ j \in \{1, ..., m_i\} \ \wedge \ p \in \{1, ..., m_k\}$$

3.2 Graphical Representation

We can represent the decision process that our ILP model involves as a graph, with the nodes corresponding to individual labels and the edges marking the association between labels belonging to correlated tasks. In Figure 2, task T_1 is correlated with task T_2 and task T_2 with task T_n. No correlation exists for pair T_1, T_n. Both nodes and edges are augmented with costs. The goal is to select a subset of connected nodes, minimizing the overall cost, given that for each group of nodes $T_1, T_2, ..., T_n$ exactly one node must be selected, and the selected nodes, representing correlated tasks, must be connected. We can see that in contrast to the pipeline approach (cf. Figure 1), no *local* decisions determine the overall assignment as the *global* distribution of costs is considered.

4 Application for NL Generation Tasks

We applied the ILP model described in the previous section to integrate different tasks in an NLG application that we describe in detail in Marciniak & Strube (2004). Our classification-based approach to language generation assumes that different types of linguistic decisions involved in the generation process can be represented in a uniform way as classification problems. The linguistic knowledge required to solve the respective classifications is then learned from a corpus annotated with both semantic and grammatical information. We have applied this framework to generating natural language route directions, e.g.:

> (a) Standing in front of the hotel (b) follow Meridian street south for about 100 meters, (c) passing the First Union Bank entrance on your right, (d) until you see the river side in front of you.

We analyze the content of such texts in terms of temporally related situations, i.e. *actions* (b), *states* (a) and *events* (c,d), denoted by individual discourse units[8]. The semantics of each discourse unit is further given by a set of attributes specifying the semantic frame and aspectual category of the profiled situation. Our corpus of semantically annotated route directions comprises 75 texts with a total number of 904 discourse units (see Marciniak & Strube (2005)). The grammatical form of the texts is modeled in terms of LTAG trees also represented as feature vectors with individual features denoting syntactic and lexical elements at both the discourse and clause levels. The generation of each discourse unit consists in assigning values to the respective features, of which the LTAG trees are then assembled. In Marciniak & Strube (2004) we implemented the generation process sequentially as a cascade of classifiers that realized incrementally the vector representation of the generated text's form, given the meaning vector as input. The classifiers handled the following eight tasks, all derived from the LTAG-based representation of the grammatical form:

$\mathbf{T_1}$: **Discourse Units Rank** is concerned with ordering discourse units at the local level, i.e. only clauses temporally related to the same *parent* clause are considered. This task is further split into a series of binary *precedence* classifications that determine the relative position of two discourse units at a time

[8]The temporal structure was represented as a tree, with discourse units as nodes.

Discourse Unit	T_3	T_4	T_5
Pass the First Union Bank ...	null	vp	bare inf.
It is necessary that you pass ...	null	np+vp	bare inf.
Passing the First Union Bank ...	null	vp	gerund
After passing ...	after	vp	gerund
After your passing ...	after	np+vp	gerund
As you pass ...	as	np+vp	fin. pres.
Until you pass ...	until	np+vp	fin. pres.
Until passing ...	until	vp	gerund

Table 1: Different realizations of tasks: Connective, Verb Form and S Exp. Rare but correct constructions are in italics.

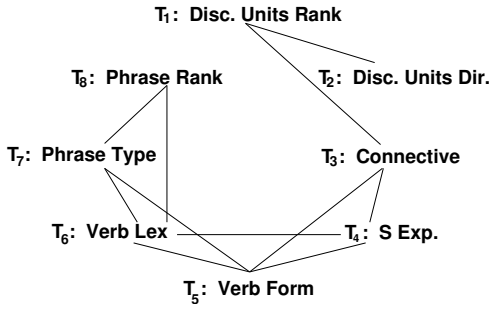


Figure 3: Correlation network for the generation tasks. Correlated tasks, are connected with lines.

(e.g. (a) *before* (c), (c) *before* (d), etc.). These partial results are later combined to determine the ordering.

$\mathbf{T_2}$: **Discourse Unit Position** specifies the position of the *child* discourse unit relative to the parent one (e.g. (a) *left of* (b), (c) *right of* (b), etc.).

$\mathbf{T_3}$: **Discourse Connective** determines the lexical form of the discourse connective (e.g. *null* in (a), *until* in (d)).

$\mathbf{T_4}$: **S Expansion** specifies whether a given discourse unit would be realized as a clause with the explicit subject (i.e. np+vp *expansion* of the root S node in a clause) (e.g. (d)) or not (e.g. (a), (b)).

$\mathbf{T_5}$: **Verb Form** determines the form of the main verb in a clause (e.g. *gerund* in (a), (c), *bare infinitive* in (b), *finite present* in (d)).

$\mathbf{T_6}$: **Verb Lexicalization** provides the lexical form of the main verb (e.g. *stand, follow, pass, etc.*).

$\mathbf{T_7}$: **Phrase Type** determines for each verb argument in a clause its syntactic realization as a *noun phrase*, *prepositional phrase* or a *particle*.

$\mathbf{T_8}$: **Phrase Rank** determines the ordering of verb arguments within a clause. As in $\mathbf{T_1}$ this task is split into a number binary classifications.

To apply the LP model to the generation problem discussed above, we first determined which pairs of tasks are correlated. The obtained network (Figure 3) is consistent with traditional analyses of the linguistic structure in terms of adjacent but separate levels: *discourse*, *clause*, *phrase*. Only a few correlations extend over level boundaries and tasks within those levels are correlated. As an example consider three interrelated tasks: Connective, S Exp. and Verb Form and their different realizations presented in Table 1. Apparently different realization of any of these tasks can affect the overall meaning of a discourse unit or its stylistics. It can also be seen that only certain combinations of different forms are allowed in the given semantic context. We can conclude that for such groups of tasks sequential processing may fail to deliver an optimal assignment.

5 Experiments and Results

In order to evaluate our approach we conducted experiments with two implementations of the ILP model and two different pipelines (*presented below*). Each system takes as input a tree structure, representing the temporal structure of the text. Individual nodes correspond to single discourse units and their semantic content is given by respective feature vectors. Generation occurs in a number of stages, during which individual discourse units are realized.

5.1 Implemented Systems

We used the ILP model described in Section 3 to build two generation systems. To obtain assignment costs, both systems get a probability distribution for each task from basic classifiers trained on the training data. To calculate the separation costs, modeling the stochastic constraints on the co-occurrence of labels, we considered correlated tasks only (cf. Figure 3) and applied two calculation methods, which resulted in two different system implementations.

In ILP1, for each pair of tasks we computed the joint distribution of the respective labels considering all discourse units in the training data before the actual input was known. Such obtained joint distributions were used for generating all discourse units from the test data. An example matrix with joint distribution for selected labels of tasks Connective and Verb Form is given in Table 2. An advantage of this

null	*and*	*as*	*after*	*until*	T_3 Connective / T_5 Verb Form
0.40	0.18	0	0	0	*bare_inf*
0	0	0	0.04	0.01	*gerund*
0.05	0.01	0.06	0.03	0.06	*fin_pres*
0.06	0.05	0	0	0	*will_inf*

Table 2: Joint distribution matrix for selected labels of tasks Connective *(horizontal)* and Verb Form *(vertical)*, computed for all discourse units in a corpus.

null	*and*	*as*	*after*	*until*	T_3 Connective / T_5 Verb Form
0.13	0.02	0	0	0	*bare_inf*
0	0	0	0	0	*gerund*
0	0	0.05	0.02	0.27	*fin_pres*
0.36	0.13	0	0	0	*will_inf*

Table 3: Joint distribution matrix for tasks Connective and Verb Form, considering only discourse units similar to (c): *until you see the river side in front of you*, at *Phi*-threshold ≥ 0.8.

approach is that the computation can be done in an *offline* mode and has no impact on the run-time.

In ILP2, the joint distribution for a pair of tasks was calculated at run-time, i.e. only after the actual input had been known. This time we did not consider all discourse units in the training data, but only those whose meaning, represented as a feature vector was *similar* to the meaning vector of the input discourse unit. As a similarity metric we used the *Phi* coefficient[9], and set the similarity threshold at 0.8. As can be seen from Table 3, the probability distribution computed in this way is better suited to the specific semantic context. This is especially important if the available corpus is small and the frequency of certain pairs of labels might be too low to have a significant impact on the final assignment.

As a baseline we implemented two pipeline systems. In the first one we used the ordering of tasks most closely resembling the conventional NLG pipeline (see Figure 4). Individual classifiers had access to both the semantic features, and those output by the previous modules. To train the classifiers, the *correct* feature values were extracted from the training data and during testing the *generated*, and hence possibly erroneous, values were taken. In the other pipeline system we wanted to minimize the error-propagation effect and placed the tasks in the order of *decreasing* accuracy. To determine the ordering of tasks we applied the following procedure: the classifier with the highest baseline accuracy was selected as the first one. The remaining classifiers were trained and tested again, but this time they had access to the additional feature. Again, the classifier with the highest accuracy was selected and the procedure was repeated until all classifiers were ordered.

5.2 Evaluation

We evaluated our system using *leave-one-out* cross-validation, i.e. for all texts in the corpus, each text was used once for testing, and the remaining texts provided the training data. To solve individual classification tasks we used the decision tree learner *C4.5* in the pipeline systems and the *Naive Bayes* algorithm[10] in the ILP systems. Both learning schemes yielded highest results in the respective configurations[11]. For each task we applied a *feature selection* procedure (cf. Kohavi & John (1997)) to determine which *semantic* features should be taken as the input by the respective basic classifiers[12]. We started with an empty feature set, and then performed experiments checking classification accuracy with only one new feature at a time. The feature that scored highest was then added to the feature set and the whole procedure was repeated iteratively until no performance improvement took place, or no more features were left.

To evaluate individual tasks we applied two metrics: accuracy, calculated as the proportion of correct classifications to the total number of instances, and the κ statistic, which corrects for the proportion of classifications that might occur by chance[13]

[9] *Phi* is a measure of the extent of correlation between two sets of binary variables, see e.g. Edwards (1976). To represent multi-class features on a binary scale we applied *dummy coding* which transforms multi class-nominal variables to a set of dummy variables with binary values.

[10] Both implemented in the Weka machine learning software (Witten & Frank, 2000).

[11] We have found that in direct comparison *C4.5* reaches higher accuracies than *Naive Bayes* but the probability distribution that it outputs is strongly biased towards the *winning* label. In this case it is practically impossible for the ILP system to change the classifier's decision, as the costs of other labels get extremely high. Hence the more balanced probability distribution given by *Naive Bayes* can be easier *corrected* in the optimization process.

[12] I.e. trained using the semantic features only, with no access to the outputs of other tasks.

[13] Hence the κ values obtained for tasks of different *difficul-*

Tasks	Pipeline 1 Pos.	Pipeline 1 Accuracy	Pipeline 1 κ	Pipeline 2 Pos.	Pipeline 2 Accuracy	Pipeline 2 κ	ILP 1 Accuracy	ILP 1 κ	ILP 2 Accuracy	ILP 2 κ
Dis.Un. Rank	1	96.81%	90.90%	2	96.81%	90.90%	97.43%	92.66%	97.43%	92.66%
Dis.Un. Pos.	2	98.04%	89.64%	1	98.04%	89.64%	96.10%	77.19%	97.95%	89.05%
Connective	3	78.64%	60.33%	7	79.10%	61.14%	79.15%	61.22%	79.36%	61.31%
S Exp.	4	95.90%	89.45%	3	96.20%	90.17%	99.48%	98.65%	99.49%	98.65%
Verb Form	5	86.76%	77.01%	4	87.83%	78.90%	92.81%	87.60%	93.22%	88.30%
Verb Lex	6	64.58%	60.87%	8	67.40%	64.19%	75.87%	73.69%	76.08%	74.00%
Phr. Type	7	86.93%	75.07%	5	87.08%	75.36%	87.33%	76.75%	88.03%	77.17%
Phr. Rank	8	84.73%	75.24%	6	86.95%	78.65%	90.22%	84.02%	91.27%	85.72%
Phi	0.85			0.87			0.89		0.90	

Table 4: Results reached by the implemented ILP systems and two baselines. For both pipeline systems, *Pos.* stands for the position of the tasks in the pipeline.

(Siegel & Castellan, 1988). For *end-to-end* evaluation, we applied the *Phi* coefficient to measure the degree of similarity between the vector representations of the generated form and the reference form obtained from the test data. The *Phi* statistic is similar to κ as it compensates for the fact that a match between two multi-label features is more difficult to obtain than in the case of binary features. This measure tells us how well all the tasks have been solved together, which in our case amounts to generating the whole text.

The results presented in Table 4 show that the ILP systems achieved highest accuracy and κ for most tasks and reached the highest overall *Phi* score. Notice that for the three correlated tasks that we considered before, i.e. Connective, S Exp. and Verb Form, ILP2 scored noticeably higher than the pipeline systems. It is interesting to see the effect of sequential processing on the results for another group of correlated tasks, i.e. Verb Lex, Phrase Type and Phrase Rank (cf. Figure 3). Verb Lex got higher scores in Pipeline2, with outputs from both Phrase Type and Phrase Rank (see the respective pipeline positions), but the reverse effect did not occur: scores for both phrase tasks were lower in Pipeline1 when they had access to the output from Verb Lex, contrary to what we might expect. Apparently, this was due to the low accuracy for Verb Lex which caused the already mentioned error propagation[14]. This example shows well the advantage that optimization processing brings: both ILP systems reached much higher scores for all three tasks.

5.3 Technical Notes

The size of an LP model is typically expressed in the number of variables and constraints. In the model presented here it depends on the number of tasks in T, the number of possible labels for each task, and the number of correlated tasks. For n different tasks with the average of m labels, and assuming every two tasks are correlated with each other, the number of variables in the LP target functions is given by: $num(var) = n \cdot m + 1/2 \cdot n(n-1) \cdot m^2$ and the number of constraints by: $num(cons) = n + n \cdot (n-1) \cdot m$. To solve the ILP models in our system we use *lp_solve*, an efficient GNU-licence Mixed Integer Programming (MIP) solver[15], which implements the Branch-and-Bound algorithm. In our application, the models varied in size from: 557 variables and 178 constraints to 709 variables and 240 constraints, depending on the number of arguments in a sentence. Generation of a text with 23 discourse units took under 7 seconds on a two-processor 2000 MHz AMD machine.

6 Conclusions

In this paper we argued that pipeline architectures in NLP can be successfully replaced by optimization models which are better suited to handling correlated tasks. The ILP formulation that we proposed extends the classification paradigm already established in NLP and is general enough to accommodate various kinds of tasks, given the right kind of data. We applied our model in an NLG application. The results we obtained show that discrete

ties can be directly compared, which gives a clear notion how well individual tasks have been solved.

[14] Apparantly, tasks which involve lexical choice get low scores with retrieval measures as the semantic content allows typically more than one correct form

[15] http://www.geocities.com/lpsolve/

optimization eliminates some limitations of sequential processing, and we believe that it can be successfully applied in other areas of NLP. We view our work as an extension to Roth & Yih (2004) in two important aspects. We experiment with a larger number of tasks having a varying number of labels. To lower the complexity of the models, we apply correlation tests, which rule out pairs of unrelated tasks. We also use stochastic constraints, which are application-independent, and for any pair of tasks can be obtained from the data.

A similar argument against sequential modularization in NLP applications was raised by van den Bosch et al. (1998) in the context of word pronunciation learning. This mapping between words and their phonemic transcriptions traditionally assumes a number of intermediate stages such as morphological segmentation, graphemic parsing, grapheme-phoneme conversion, syllabification and stress assignment. The authors report an increase in generalization accuracy when the the modular decomposition is abandoned (i.e. the tasks of conversion to phonemes and stress assignment get conflated and the other intermediate tasks are skipped). It is interesting to note that a similar dependence on the intermediate abstraction levels is present in such applications as parsing and semantic role labelling, which both assume POS tagging and chunking as their preceding stages.

Currently we are working on a uniform data format that would allow to represent different NLP applications as multi-task optimization problems. We are planning to release a task-independent Java API that would solve such problems. We want to use this generic model for building NLP modules that traditionally are implemented sequentially.

Acknowledgements: The work presented here has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author receives a scholarship from KTF (09.001.2004).

References

Althaus, E., N. Karamanis & A. Koller (2004). Computing locally coherent discourses. In *Proceedings of the 42 Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, July 21-26, 2004, pp. 399–406.

Buchholz, S., J. Veenstra & W. Daelemans (1999). Cascaded grammatical relation assignment. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora,* College Park, Md., June 21-22, 1999, pp. 239–246.

Chekuri, C., S. Khanna, J. Naor & L. Zosin (2001). Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the 12th Annual ACM SIAM Symposium on Discrete Algorithms,* Washington, DC, pp. 109–118.

Cunningham, H., K. Humphreys, Y. Wilks & R. Gaizauskas (1997). Software infrastructure for natural language processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing* Washington, DC, March 31 - April 3, 1997, pp. 237–244.

Daelemans, W. & A. van den Bosch (1998). Rapid development of NLP modules with memory-based learning. In *Proceedings of ELSNET in Wonderland. Utrecht: ELSNET*, pp. 105–113.

Edwards, Allen, L. (1976). *An Introduction to Linear Regression and Correlation*. San Francisco, Cal.: W. H. Freeman.

Goodman, L. A. & W. H. Kruskal (1972). Measures of association for cross-classification, iv. *Journal of the American Statistical Association*, 67:415–421.

Kleinberg, J. M. & E. Tardos (2000). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM*, 49(5):616–639.

Kohavi, R. & G. H. John (1997). Wrappers for feature subset selection. *Artificial Intelligence Journal*, 97:273–324.

Marciniak, T. & M. Strube (2004). Classification-based generation using TAG. In *Proceedings of the 3rd International Conference on Natural Language Generation,* Brockenhurst, UK, 14-16 July, 2004, pp. 100–109.

Marciniak, T. & M. Strube (2005). Modeling and annotating the semantics of route directions. In *Proceedings of the 6th International Workshop on Computational Semantics,* Tilburg, The Netherlands, January 12-14, 2005, pp. 151–162.

Nemhauser, G. L. & L. A. Wolsey (1999). *Integer and combinatorial optimization*. New York, NY: Wiley.

Punyakanok, V., D. Roth, W. Yih & Z. Dav (2004). Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computational Linguistics,* Geneva, Switzerland, August 23-27, 2004, pp. 1346–1352.

Reiter, E. (1994). Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation,* Kennebunkport, Maine, pp. 160–173.

Reiter, E. & R. Dale (2000). *Building Natural Language Generation Systems*. Cambridge, UK: Cambridge University Press.

Roth, D. & W. Yih (2004). A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning,* Boston, Mass., May 2-7, 2004, pp. 1–8.

Siegel, S. & N. J. Castellan (1988). *Nonparametric Statistics for the Behavioral Sciences*. New York, NY: McGraw-Hill.

Soon, W. M., H. T. Ng & D. C. L. Lim (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

van den Bosch, A., T. Weijters & W. Daelemans (1998). Modularity in inductively-learned word pronunciation systems. In D. Powers (Ed.), *Proceedings of NeMLaP3/CoNLL98*, pp. 185–194.

Witten, I. H. & E. Frank (2000). *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, Cal.: Morgan Kaufmann.

Investigating the Effects of Selective Sampling on the Annotation Task

Ben Hachey, Beatrice Alex and **Markus Becker**
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, UK
{bhachey,v1balex,s0235256}@inf.ed.ac.uk

Abstract

We report on an active learning experiment for named entity recognition in the astronomy domain. Active learning has been shown to reduce the amount of labelled data required to train a supervised learner by selectively sampling more informative data points for human annotation. We inspect double annotation data from the same domain and quantify potential problems concerning annotators' performance. For data selectively sampled according to different selection metrics, we find lower inter-annotator agreement and higher per token annotation times. However, overall results confirm the utility of active learning.

1 Introduction

Supervised training of named entity recognition (NER) systems requires large amounts of manually annotated data. However, human annotation is typically costly and time-consuming. Active learning promises to reduce this cost by requesting only those data points for human annotation which are highly informative. Example informativity can be estimated by the degree of uncertainty of a single learner as to the correct label of a data point (Cohn et al., 1995) or in terms of the disagreement of a committee of learners (Seung et al., 1992). Active learning has been successfully applied to a variety of tasks such as document classification (McCallum and Nigam, 1998), part-of-speech tagging (Argamon-Engelson and Dagan, 1999), and parsing (Thompson et al., 1999).

We employ a committee-based method where the degree of deviation of different classifiers with respect to their analysis can tell us if an example is potentially useful. In a companion paper (Becker et al., 2005), we present active learning experiments for NER in radio-astronomical texts following this approach.[1] These experiments prove the utility of selective sampling and suggest that parameters for a new domain can be optimised in another domain for which annotated data is already available.

However there are some provisos for active learning. An important point to consider is what effect *informative* examples have on the annotators. Are these examples more difficult? Will they affect the annotators' performance in terms of accuracy? Will they affect the annotators performance in terms of time? In this paper, we explore these questions using doubly annotated data. We find that selective sampling does have an adverse effect on annotator accuracy and efficiency.

In section 2, we present standard active learning results showing that good performance can be achieved using fewer examples than random sampling. Then, in section 3, we address the questions above, looking at the relationship between inter-annotator agreement and annotation time and the examples that are selected by active learning. Finally, section 4 presents conclusions and future work.

[1] Please refer to the companion paper for details of the selective sampling approach with experimental adaptation results as well as more information about the corpus of radio-astronomical abstracts.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 144–151, Ann Arbor, June 2005.

2 Bootstrapping NER

The work reported here was carried out in order to assess methods of porting a statistical NER system to a new domain. We started with a NER system trained on biomedical literature and built a new system to identify four novel entities in abstracts from astronomy articles. This section introduces the Astronomy Bootstrapping Corpus (ABC) which was developed for the task, describes our active learning approach to bootstrapping, and gives a brief overview of the experiments.

2.1 The Astronomy Bootstrapping Corpus

The ABC corpus consists of abstracts of radio astronomical papers from the NASA Astrophysics Data System archive[2], a digital library for physics, astrophysics, and instrumentation. Abstracts were extracted from the years 1997-2003 that matched the query "quasar AND line". A set of 50 abstracts from the year 2002 were annotated as seed material and 159 abstracts from 2003 were annotated as testing material. A further 778 abstracts from the years 1997-2001 were provided as an unannotated pool for bootstrapping. On average, these abstracts contain 10 sentences with a length of 30 tokens. The annotation marks up four entity types:

Instrument-name (IN) Names of telescopes and other measurement instruments, e.g. *Superconducting Tunnel Junction (STJ) camera, Plateau de Bure Interferometer, Chandra, XMM-Newton Reflection Grating Spectrometer (RGS), Hubble Space Telescope.*

Source-name (SN) Names of celestial objects, e.g. *NGC 7603, 3C 273, BRI 1335-0417, SDSSp J104433.04-012502.2, PC0953+ 4749.*

Source-type (ST) Types of objects, e.g. *Type II Supernovae (SNe II), radio-loud quasar, type 2 QSO, starburst galaxies, low-luminosity AGNs.*

Spectral-feature (SF) Features that can be pointed to on a spectrum, e.g. *Mg II emission, broad emission lines, radio continuum emission at 1.47 GHz, CO ladder from (2-1) up to (7-6), non-LTE line.*

[2] `http://adsabs.harvard.edu/preprint_service.html`

The seed and test data sets were annotated by two astrophysics PhD students. In addition, they annotated 1000 randomly sampled sentences from the pool to provide a random baseline for active learning. These sentences were doubly annotated and adjudicated and form the basis for our calculations in section 3.

2.2 Inter-Annotator Agreement

In order to ensure consistency in annotation projects, corpora are often annotated by more than one annotator, e.g. in the annotation of the Penn Treebank (Marcus et al., 1994). In these cases, inter-annotator agreement is frequently reported between different annotated versions of a corpus as an indicator for the difficulty of the annotation task. For example, Brants (2000) reports inter-annotator agreement in terms of accuracy and f-score for the annotation of the German NEGRA treebank.

Evaluation metrics for named entity recognition are standardly reported as accuracy on the token level, and as f-score on the phrasal level, e.g. Sang (2002), where token level annotation refers to the B-I-O coding scheme.[3] Likewise, we will use accuracy to report inter-annotator agreement on the token level, and f-score for the phrase level. We may arbitrarily assign one annotator's data as the gold standard, since both accuracy and f-score are symmetric with respect to the test and gold set. To see why this is the case, note that accuracy can simply be defined as the ratio of the number of tokens on which the annotators agree over the total number of tokens. Also the f-score is symmetric, since recall(A,B) = precision(B,A) and (balanced) f-score is the harmonic mean of recall and precision (Brants, 2000). The pairwise f-score for the ABC corpus is 85.52 (accuracy of 97.15) with class information and 86.15 (accuracy of 97.28) without class information. The results in later sections will be reported using this pairwise f-score for measuring agreement.

For NER, it is also common to compare an annotator's tagged document to the final, reconciled version of the document, e.g. Robinson et al. (1999) and Strassel et al. (2003). The inter-annotator f-score agreement calculated this way for MUC-7 and Hub 4 was measured at 97 and 98 respectively. The

[3] B-X marks the beginning of a phrase of type X, I-X denotes the continuation of an X phrase, and O a non-phrasal token.

doubly annotated data for the ABC corpus was resolved by the original annotators in the presence of an astronomy adjudicator (senior academic staff) and a computational linguist. This approach gives an f-score of 91.89 (accuracy of 98.43) with class information for the ABC corpus. Without class information, we get an f-score of 92.22 (accuracy of 98.49), indicating that most of our errors are due to boundary problems. These numbers suggest that our task is more difficult than the generic NER tasks from the MUC and HUB evaluations.

Another common agreement metric is the kappa coefficient which normalises token level accuracy by chance, e.g. Carletta et al. (1997). This metric showed that the human annotators distinguish the four categories with a reproducibility of K=.925 (N=44775, k=2; where K is the kappa coefficient, N is the number of tokens and k is the number of annotators).

2.3 Active Learning

We have already mentioned that there are two main approaches in the literature to assessing the informativity of an example: the degree of uncertainty of a single learner and the disagreement between a committee of learners. For the current work, we employ query-by-committee (QBC). We use a conditional Markov model (CMM) tagger (Klein et al., 2003; Finkel et al., 2005) to train two different models on the same data by splitting the feature set. In this section we discuss several parameters of this approach for the current task.

Level of annotation For the manual annotation of named entity examples, we needed to decide on the level of granularity. The question arises of what constitutes an example that will be submitted to the annotators. Possible levels include the document level, the sentence level and the token level. The most fine-grained annotation would certainly be on the token level. However, it seems unnatural for the annotator to label individual tokens. Furthermore, our machine learning tool models sequences at the sentence level and does not allow to mix unannotated tokens with annotated ones. At the other extreme, one may submit an entire document for annotation. A possible disadvantage is that a document with some interesting parts may well contain large portions with redundant, already known structures for which knowing the manual annotation may not be very useful. In the given setting, we decided that the best granularity is the sentence.

Sample Selection Metric There are a variety of metrics that could be used to quantify the degree of deviation between classifiers in a committee (e.g. KL-divergence, information radius, f-measure). The work reported here uses two sentence-level metrics based on KL-divergence and one based on f-measure.

KL-divergence has been used for active learning to quantify the disagreement of classifiers over the probability distribution of output labels (McCallum and Nigam, 1998; Jones et al., 2003). It measures the divergence between two probability distributions p and q over the same event space χ:

$$D(p||q) = \sum_{x \in \chi} p(x) \log \frac{p(x)}{q(x)} \quad (1)$$

KL-divergence is a non-negative metric. It is zero for identical distributions; the more different the two distributions, the higher the KL-divergence. Intuitively, a high KL-divergence score indicates an informative data point. However, in the current formulation, KL-divergence only relates to individual tokens. In order to turn this into a sentence score, we need to combine the individual KL-divergences for the tokens within a sentence into one single score. We employed mean and max.

The *f-complement* has been suggested for active learning in the context of NP chunking as a structural comparison between the different analyses of a committee (Ngai and Yarowsky, 2000). It is the pairwise f-measure comparison between the multiple analyses for a given sentence:

$$f^{\mathcal{M}}_{comp} = \frac{1}{2} \sum_{M,M' \in \mathcal{M}} (1 - F_1(M(t), M'(t))) \quad (2)$$

where F_1 is the balanced f-measure of $M(t)$ and $M'(t)$, the preferred analyses of data point t according to different members M, M' of ensemble $\mathcal{M}$. We take the complement so that it is oriented the same as KL-divergence with high values indicating high disagreement. This is equivalent to taking the inter-annotator agreement between $|\mathcal{M}|$ classifiers.

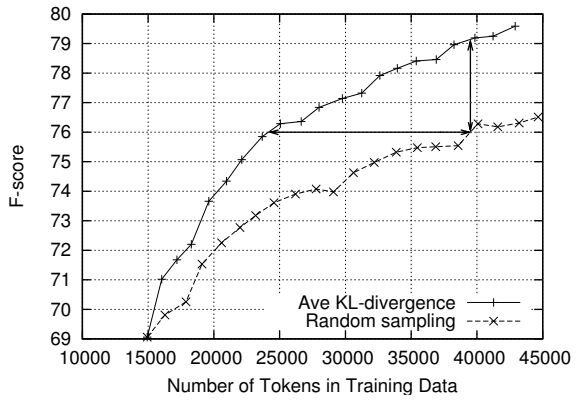


Figure 1: Learning curve of the real AL experiment.

2.4 Experiments

To tune the active learning parameters discussed in section 2.3, we ran detailed simulated experiments on the named entity data from the BioNLP shared task of the COLING 2004 International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (Kim et al., 2004). These results are treated in detail in the companion paper (Becker et al., 2005).

We used the CMM tagger to train two different models by splitting the feature set to give multiple views of the same data. The feature set was handcrafted such that it comprises different views while empirically ensuring that performance is sufficiently similar. On the basis of the findings of the simulation experiments we set up the real active learning annotation experiment using: average KL-divergence as the selection metric and a feature split that divides the full feature set roughly into features of words and features derived from external resources. As smaller batch sizes require more retraining iterations and larger batch sizes increase the amount of annotation necessary at each round and could lead to unnecessary strain for the annotators, we settled on a batch size of 50 sentences for the real AL experiment as a compromise between computational cost and work load for the annotator.

We developed an active annotation tool and ran real annotation experiments on the astronomy abstracts described in section 2.1. The tool was given to the same astronomy PhD students for annotation who were responsible for the seed and test data. The learning curve for selective sampling is plotted in figure 1.[4] The randomly sampled data was doubly annotated and the learning curve is averaged between the two annotators.

Comparing the selective sampling performance to the baseline, we confirm that active learning provides a significant reduction in the number of examples that need annotating. In fact, the random curve reaches an f-score of 76 after approximately 39000 tokens have been annotated while the selective sampling curve reaches this level of performance after only $\approx$ 24000 tokens. This represents a substantial reduction in tokens annotated of 38.5%. In addition, at 39000 tokens, selectively sampling offers an error reduction of 21.4% with a 3 point improvement in f-score.

3 Evaluating Selective Sampling

Standardly, the evaluation of active learning methods and the comparison of sample selection metrics draws on experiments over gold-standard annotated corpora, where a set of annotated data is at our disposal, e.g. McCallum and Nigam (1998), Osborne and Baldridge (2004). This assumes implicitly that annotators will always produce gold-standard quality annotations, which is typically not the case, as we discussed in Section 2.2. What is more, we speculate that annotators might have an even higher error rate on the supposedly more informative, but possibly also more difficult examples. However, this would not be reflected in the carefully annotated and verified examples of a gold standard corpus. In the following analysis, we leverage information from doubly annotated data to explore the effects on annotation of selectively sampled examples.

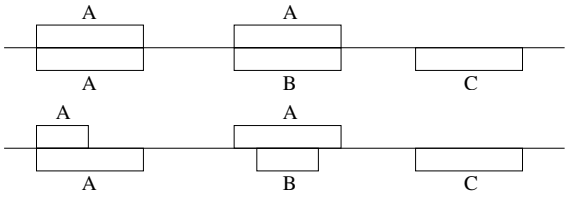
To evaluate the practicality and usefulness of active learning as a generally applicable methodology, it is desirable to be able to observe the behaviour of the annotators. In this section, we will report on the evaluation of various subsets of the doubly annotated portion of the ABC corpus comprising 1000 sentences, which we sample according to a sample selection metric. That is, examples are added to the subsets according to the sample selection metric, selecting those with higher disagreement first. This allows us to trace changes in inter-annotator agree-

[4] Learning curves reflect the performance on the test set using the full feature set.

ment between the full corpus and selected subsets thereof. Also, we will inspect timing information. This novel methodology allows us to experiment with different sample selection metrics without having to repeat the actual time and resource intensive annotation.

3.1 Error Analysis

To investigate the types of classification errors, it is common to set up a confusion matrix. One approach is to do this at the token level. However, we are dealing with phrases and our analysis should reflect that. Thus we devised a method for constructing a confusion matrix based on phrasal alignment. These confusion matrices are constructed by giving a double count for each phrase that has matching boundaries and a single count for each phrase that does not have matching boundaries. To illustrate, consider the following sentences–annotated with phrases A, B, and C for annotator 1 on top and annotator 2 on bottom–as sentence 1 and sentence 2 respectively:



Sentence 1 will get a count of 2 for A/A and for A/B and a count of 1 for O/C, while sentence 2 will get 2 counts of A/O, and 1 count each of O/A, O/B, and O/C. Table 1 contains confusion matrices for the first 100 sentences sorted by averaged KL-divergence and for the full set of 1000 random sentences from the pool data. (Note that these confusion matrices contain percentages instead of raw counts so they can be directly compared.)

We can make some interesting observations looking at these phrasal confusion matrices. The main effect we observed is the same as was suggested by the f-score inter-annotator agreement errors in section 2.1. Specifically, looking at the full random set of 1000 sentences, almost all errors (Where $*$ is any entity phrase type, $\frac{*/O + O/* \; errors}{all \; errors}$ = 95.43%) are due to problems with phrase boundaries. Comparing the full random set to the 100 sentences with the highest averaged KL-divergence, we can see that this is even more the case for the sub-set of 100 sentences (97.43%). Therefore, we can observe that

100:

A1 \ A2	IN	SN	ST	SF	O
IN	12.0	0.0	0.0	0.0	0.4
SN	0.0	10.4	0.0	0.0	0.4
ST	0.0	0.4	30.3	0.0	1.0
SF	0.0	0.0	0.0	31.1	3.9
O	0.2	0.4	2.9	6.4	—

1000:

A1 \ A2	IN	SN	ST	SF	O
IN	9.4	0.0	0.0	0.0	0.3
SN	0.0	10.1	0.2	0.1	0.3
ST	0.0	0.1	41.9	0.1	1.6
SF	0.0	0.0	0.1	25.1	3.0
O	0.3	0.2	2.4	4.8	—

Table 1: Phrasal confusion matrices for document sub-set of 100 sentences sorted by average KL-divergence and for full random document sub-set of 1000 sentences (A1: Annotator 1, A2: Annotator 2).

Entity	100	1000
`Instrument-name`	12.4%	9.7%
`Source-name`	10.8%	10.7%
`Source-type`	31.7%	43.7%
`Spectral-feature`	35.0%	28.2%
O	9.9%	7.7%

Table 2: Normalised distributions of agreed entity annotations.

there is a tendency for the averaged KL-divergence selection metric to choose sentences where phrase boundary identification is difficult.

Furthermore, comparing the confusion matrices for 100 sentences and for the full set of 1000 shows that sentences containing less common entity types tend to be selected first while sentences containing the most common entity types are dispreferred. Table 2 contains the marginal distribution for annotator 1 (A1) from the confusion matrices for the ordered sub-set of 100 and for the full random set of 1000 sentences. So, for example, the sorted sub-set contains 12.4% `Instrument-name` annotations (the least common entity type) while the full set contains 9.7%. And, 31.7% of agreed entity annotations in the first sub-set of 100 are `Source-type` (the most common entity type), whereas the propor-

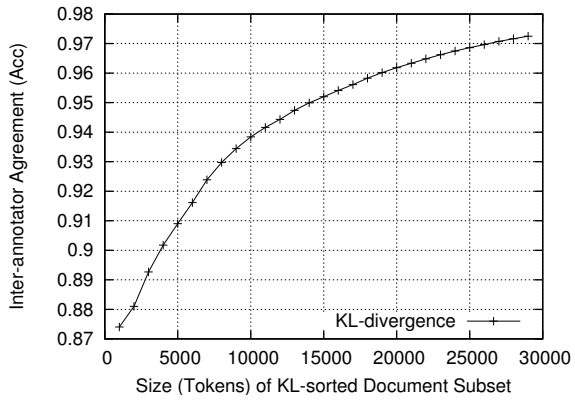


Figure 2: Raw agreement plotted against KL-sorted document subsets.

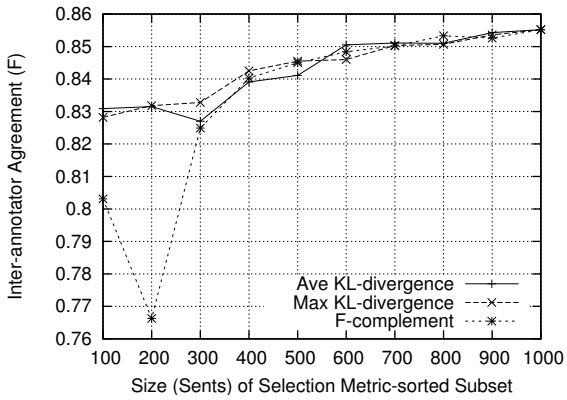


Figure 3: Human disagreement plotted against selection metric-sorted document subsets.

tion of agreed `Source-type` annotations in the full random set is 43.7%. Looking at the O row, we also observe that sentences with difficult phrases are preferred. A similar effect can be observed in the marginals for annotator 2.

3.2 Annotator Performance

So far, the behaviour we have observed is what you would expect from selective sampling; there is a marked improvement in terms of cost and error rate reduction over random sampling. However, selective sampling raises questions of cognitive load and the quality of annotation. In the following section we investigate the relationship between informativity, inter-annotator agreement, and annotation time.

While reusability of selective samples for other learning algorithms has been explored (Baldridge and Osborne, 2004), no effort has been made to quantify the effect of selective sampling on annotator performance. We concentrate first on the question: *Are informative examples more difficult to annotate?* One way to quantify this effect is to look at the correlation between human agreement and the token-level KL-divergence. The Pearson correlation coefficient indicates the degree to which two variables are related. It ranges between -1 and 1, where 1 means perfectly positive correlation, and -1 perfectly negative correlation. A value of 0 indicates no correlation. The Pearson correlation coefficient on all tokens gives a very weak correlation coefficient of -0.009.[5] However, this includes many trivial tokens which are easily identified as being outside an entity phrase. If we look just at tokens that at least one of the annotators posits as being part of an entity phrase, we observe a larger effect with a Pearson correlation coefficient of -0.120, indicating that agreement tends to be low when KL-divergence is high. Figure 2 illustrates this effect even more dramatically. Here we plot accuracy against token subsets of size $1000, 2000, .., N$ where tokens are added to the subsets according to their KL-divergence, selecting those with the highest values first. This demonstrates clearly that tokens with higher KL-divergence have lower inter-annotator agreement.

However, as discussed in sections 2.3 and 2.4, we decided on sentences as the preferred annotation level. Therefore, it is important to explore these relationships at the sentence level as well. Again, we start by looking at the Pearson correlation coefficient between f-score inter-annotator agreement (as described in section 2.1) and our active learning selection metrics:

	Ave KL	Max KL	1-F
All Tokens	-0.090	-0.145	-0.143
O Removed	-0.042	-0.092	-0.101

Here *O Removed* means that sentences are removed for which the annotators agree that there are no entity phrases (i.e. all tokens are labelled as being outside an entity phrase). This shows a relation-

[5] In order to make this calculation, we give token-level agreement a numeric representation by assigning 1 to tokens on which the annotators agree and 0 to tokens on which they disagree.

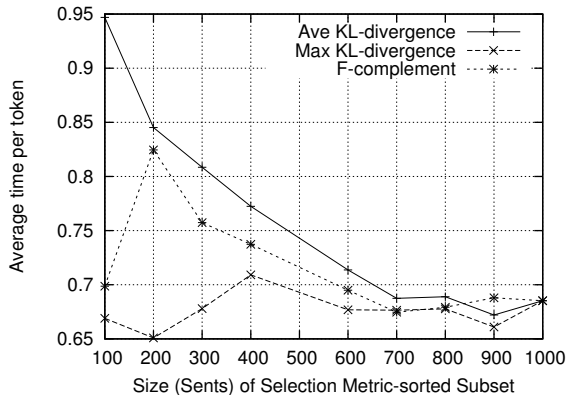


Figure 4: Annotation time plotted against selection metric-sorted document subsets.

ship very similar to what we observed at the token level: a negative correlation indicating that agreement is low when KL-divergence is high. Again, the effect of selecting informative examples is better illustrated with a plot. Figure 3 plots f-score agreement against sentence subsets sorted by our sentence level selection metrics. Lower agreement at the left of these plots indicates that the more informative examples according to our selection metrics are more difficult to annotate.

So, active learning makes the annotation more difficult. But, this raises a further question: *What effect do more difficult examples have on annotation time?* To investigate this, we once again start by looking at the Pearson correlation coefficient, this time between the annotation time and our selection metrics. However, as our sentence-level selection metrics affect the length of sentences selected, we normalise sentence-level annotation times by sentence length:

	Ave KL	Max KL	1-F
All Tokens	0.157	−0.009	0.082
O Removed	0.216	−0.007	0.106

Here we see a small positive correlations for averaged KL-divergence and f-complement indicating that sentences that score higher according to our selection metrics do generally take longer to annotate. Again, we can visualise this effect better by plotting average time against KL-sorted subsets (Figure 4). This demonstrates that sentences preferred by our selection metrics generally take longer to annotate.

4 Conclusions and Future Work

We have presented active learning experiments in a novel NER domain and investigated negative side effects. We investigated the relationship between informativity of an example, as determined by selective sampling metrics, and inter-annotator agreement. This effect has been quantified using the Pearson correlation coefficient and visualised using plots that illustrate the difficulty and time-intensiveness of examples chosen first by selective sampling. These measurements clearly demonstrate that selectively sampled examples are in fact more difficult to annotate. And, while sentence length and entities per sentence are somewhat confounding factors, we have also shown that selective sampling of informative examples appears to increase the time spent on individual examples.

High quality annotation is important for building accurate models and for reusability. While annotation quality suffers for selectively sampled examples, selective sampling nevertheless provided a dramatic cost reduction of 38.5% in a real annotation experiment, demonstrating the utility of active learning for bootstrapping NER in a new domain.

In future work, we will perform further investigations of the cost of resolving annotations for selectively sampled examples. And, in related work, we will use timing information to assess token, entity and sentence cost metrics for annotation. This should also lead to a better understanding of the relationship between timing information and sentence length for different selection metrics.

Acknowledgements

The work reported here, including the related development of the astronomy bootstrapping corpus and the active learning tools, were supported by Edinburgh-Stanford Link Grant (R36759) as part of the SEER project. We are very grateful for the time and resources invested in corpus preparation by our collaborators in the Institute for Astronomy, University of Edinburgh: Rachel Dowsett, Olivia Johnson and Bob Mann. We are also grateful to Melissa Kronenthal and Jean Carletta for help collecting data.

References

Shlomo Argamon-Engelson and Ido Dagan. 1999. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360.

Jason Baldridge and Miles Osborne. 2004. Ensemble-based active learning for parse selection. In *Proceedings of the 5th Conference of the North American Chapter of the Association for Computational Linguistics*.

Markus Becker, Ben Hachey, Beatrice Alex, and Claire Grover. 2005. Optimising selective sampling for bootstrapping named entity recognition. In *ICML-2005 Workshop on Learning with Multiple Views*.

Thorsten Brants. 2000. Inter-annotator agreement for a German newspaper corpus. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*.

Jean Carletta, Amy Isard, Stephen Isard, Jacqueline C. Kowtko, Gwyneth Doherty-Sneddon, and Anne H. Anderson. 1997. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23(1):13–31.

David. A. Cohn, Zoubin. Ghahramani, and Michael. I. Jordan. 1995. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

Jenny Finkel, Shipra Dingare, Christopher Manning, Beatrice Alex Malvina Nissim, and Claire Grover. 2005. Exploring the boundaries: Gene and protein identification in biomedical text. *BMC Bioinformatics*. In press.

Rosie Jones, Rayid Ghani, Tom Mitchell, and Ellen Riloff. 2003. Active learning with multiple view feature sets. In *ECML 2003 Workshop on Adaptive Text Extraction and Mining*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the COLING 2004 International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. 2003. Named entity recognition with character-level models. In *Proceedings the Seventh Conference on Natural Language Learning*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Andrew McCallum and Kamal Nigam. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*.

Patricia Robinson, Erica Brown, John Burger, Nancy Chinchor, Aaron Douthat, Lisa Ferro, and Lynette Hirschman. 1999. Overview: Information extraction from broadcast news. In *Proceedings DARPA Broadcast News Workshop*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 2002 Conference on Computational Natural Language Learning*.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Computational Learning Theory*.

Stephanie Strassel, Alexis Mitchell, and Shudong Huang. 2003. Multilingual resources for entity extraction. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning*.

Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling

Xavier Carreras and **Lluís Màrquez**
TALP Research Centre
Technical University of Catalonia (UPC)
{carreras,lluism}@lsi.upc.edu

Abstract

In this paper we describe the CoNLL-2005 shared task on Semantic Role Labeling. We introduce the specification and goals of the task, describe the data sets and evaluation methods, and present a general overview of the 19 systems that have contributed to the task, providing a comparative description and results.

1 Introduction

In the few last years there has been an increasing interest in shallow semantic parsing of natural language, which is becoming an important component in all kind of NLP applications. As a particular case, Semantic Role Labeling (SRL) is currently a well-defined task with a substantial body of work and comparative evaluation. Given a sentence, the task consists of analyzing the propositions expressed by some target verbs of the sentence. In particular, for each target verb all the constituents in the sentence which fill a semantic role of the verb have to be recognized. Typical semantic arguments include Agent, Patient, Instrument, etc. and also adjuncts such as Locative, Temporal, Manner, Cause, etc.

Last year, the CoNLL-2004 shared task aimed at evaluating machine learning SRL systems based only on partial syntactic information. In (Carreras and Màrquez, 2004) one may find a detailed review of the task and also a brief state-of-the-art on SRL previous to 2004. Ten systems contributed to the task, which was evaluated using the PropBank corpus (Palmer et al., 2005). The best results were around 70 in F_1 measure. Though not directly comparable, these figures are substantially lower than the best results published up to date using full parsing as input information (F_1 slightly over 79). In addition to the CoNLL-2004 shared task, another evaluation exercise was conducted in the Senseval-3 workshop (Litkowski, 2004). Eight systems relying on full parsing information were evaluated in that event using the FrameNet corpus (Fillmore et al., 2001). From the point of view of learning architectures and study of feature relevance, it is also worth mentioning the following recent works (Punyakanok et al., 2004; Moschitti, 2004; Xue and Palmer, 2004; Pradhan et al., 2005a).

Following last year's initiative, the CoNLL-2005 shared task[1] will concern again the recognition of semantic roles for the English language. Compared to the shared task of CoNLL-2004, the novelties introduced in the 2005 edition are:

- Aiming at evaluating the contribution of full parsing in SRL, the complete syntactic trees given by two alternative parsers have been provided as input information for the task. The rest of input information does not vary and corresponds to the levels of processing treated in the previous editions of the CoNLL shared task, i.e., words, PoS tags, base chunks, clauses, and named entities.

- The training corpus has been substantially enlarged. This allows to test the scalability of

[1] The official CoNLL-2005 shared task web page, including data, software and systems' outputs, is available at http://www.lsi.upc.edu/∼srlconll.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 152–164, Ann Arbor, June 2005.

learning-based SRL systems to big datasets and to compute learning curves to see how much data is necessary to train. Again, we concentrate on the PropBank corpus (Palmer et al., 2005), which is the Wall Street Journal part of the Penn TreeBank corpus enriched with predicate–argument structures.

- In order to test the robustness of the presented systems, a cross-corpora evaluation is performed using a fresh test set from the Brown corpus.

Regarding evaluation, two different settings were devised depending if the systems use the information strictly contained in the training data (*closed challenge*) or they make use of external sources of information and/or tools (*open challenge*). The closed setting allows to compare systems under strict conditions, while the open setting aimed at exploring the contributions of other sources of information and the limits of the current learning-based systems on the SRL task. At the end, all 19 systems took part in the closed challenge and none of them in the open challenge.

The rest of the paper is organized as follows. Section 2 describes the general setting of the task. Section 3 provides a detailed description of training, development and test data. Participant systems are described and compared in section 4. In particular, information about learning techniques, SRL strategies, and feature development is provided, together with performance results on the development and test sets. Finally, section 5 concludes.

2 Task Description

As in the 2004 edition, the goal of the task was to develop a machine learning system to recognize arguments of verbs in a sentence, and label them with their semantic role. A verb and its set of arguments form a *proposition* in the sentence, and typically, a sentence contains a number of propositions.

There are two properties that characterize the structure of the arguments in a proposition. First, arguments do not overlap, and are organized sequentially. Second, an argument may appear split into a number of non-contiguous phrases. For instance, in the sentence "[$_{\text{A1}}$ The apple], said John, [$_{\text{C–A1}}$ is on the table]", the utterance argument (labeled with type `A1`) appears split into two phrases. Thus, there is a set of non-overlapping arguments labeled with semantic roles associated with each proposition. The set of arguments of a proposition can be seen as a chunking of the sentence, in which chunks are parts of the semantic roles of the proposition predicate.

In practice, number of *target verbs* are marked in a sentence, each governing one proposition. A system has to recognize and label the arguments of each target verb. To support the role labeling task, sentences contain input annotations, that consist of syntactic information and named entities. Section 3 describes in more detail the annotations of the data.

2.1 Evaluation

Evaluation is performed on a collection of unseen test sentences, that are marked with target verbs and contain only predicted input annotations.

A system is evaluated with respect to *precision*, *recall* and the F_1 measure of the predicted arguments. *Precision* (p) is the proportion of arguments predicted by a system which are correct. *Recall* (r) is the proportion of correct arguments which are predicted by a system. Finally, the F_1 measure computes the harmonic mean of precision and recall, and is the final measure to compare the performance of systems. It is formulated as: $\text{F}_{\beta=1} = 2pr/(p+r)$.

For an argument to be correctly recognized, the words spanning the argument as well as its semantic role have to be correct. [2]

As an exceptional case, the verb argument of each proposition is excluded from the evaluation. This argument is the lexicalization of the predicate of the proposition. Most of the time, the verb corresponds to the target verb of the proposition, which is provided as input, and only in few cases the verb participant spans more words than the target verb. Except for non-trivial cases, this situation makes the verb fairly easy to identify and, since there is one verb with each proposition, evaluating its recognition over-estimates the overall performance of a system. For this reason, the verb argument is excluded from evaluation.

[2] The `srl-eval.pl` program is the official program to evaluate the performance of a system. It is available at the Shared Task web page.

```
And           CC      *   (S*      (S*         *     -        (AM-DIS*)  (AM-DIS*)
to            TO   (VP*   (S*   (S(VP*         *     -               *   (AM-PNC*
attract       VB      *)    *     (VP*         *     attract       (V*)         *
younger       JJR  (NP*     *     (NP*         *     -            (A1*          *
listeners     NNS     *)    *)       *))))     *     -               *)         *)
,             ,       *     *        *         *     -               *          *
Radio         NNP  (NP*     *     (NP*     (ORG*     -            (A0*       (A0*
Free          NNP     *     *        *         *     -               *          *
Europe        NNP     *)    *        *)        *)    -               *)         *)
intersperses  VBZ  (VP*)    *     (VP*         *     intersperse     *        (V*)
the           DT   (NP*     *  (NP(NP*         *     -               *       (A1*
latest        JJS     *)    *        *)        *     -               *          *
in            IN   (PP*)    *     (PP*         *     -               *          *
Western       JJ   (NP*     *     (NP*    (MISC*)    -               *          *
rock          NN      *     *        *         *     -               *          *
groups        NNS     *)    *        *))))     *     -               *          *)
.             .       *     *)       *)        *     -               *          *
```

Figure 1: An example of an annotated sentence, in columns. Input consists of words (1st column), PoS tags (2nd), base chunks (3rd), clauses (4th), full syntactic tree (5th) and named entities (6th). The 7th column marks target verbs, and their propositions are found in remaining columns. According to the PropBank Frames, for `attract` (8th), the `A0` annotates the attractor, and the `A1` the thing attracted; for `intersperse` (9th), `A0` is the arranger, and `A1` the entity interspersed.

2.2 Closed Challenge Setting

The organization provided training, development and test sets derived from the standard sections of the Penn TreeBank (Marcus et al., 1993) and PropBank (Palmer et al., 2005) corpora.

In the closed challenge, systems have to be built strictly with information contained in the training sections of the TreeBank and PropBank. Since this collection contains the gold reference annotations of both syntactic and predicate-argument structures, the closed challenge allows: (1) to make use of any preprocessing system strictly developed within this setting, and (2) to learn from scratch any annotation that is contained in the data. To support the former, the organization provided the output of state-of-the-art syntactic preprocessors, described in Section 3.

The development set is used to tune the parameters of a system. The gold reference annotations are also available in this set, but only to evaluate the performance of different parametrizations of a system, and select the optimal one. Finally, the test set is used to evaluate the performance of a system. It is only allowed to use predicted annotations in this set.

Since all systems in this setting have had access to the same training and development data, the evaluation results on the test obtained by different systems are comparable in a fair manner.

3 Data

The data consists of sections of the Wall Street Journal part of the Penn TreeBank (Marcus et al., 1993), with information on predicate-argument structures extracted from the PropBank corpus (Palmer et al., 2005). In this edition of the CoNLL shared task, we followed the standard partition used in syntactic parsing: sections 02-21 for training, section 24 for development, and section 23 for test. In addition, the test set of the shared task includes three sections of the Brown corpus (namely, ck01-03). The predicate-argument annotations of the latter test material were kindly provided by the PropBank team, and are very valuable, as they allow to evaluate learning systems on a portion of data that comes from a different source than training.

We first describe the annotations related to argument structures. Then, we describe the preprocessing systems that have been selected to predict the input part of the data. Figure 1 shows an example of a fully-annotated sentence.

3.1 PropBank

The Proposition Bank (PropBank) (Palmer et al., 2005) annotates the Penn TreeBank with verb argument structure. The semantic roles covered by PropBank are the following:

- **Numbered arguments (A0–A5, AA):** Arguments defining verb-specific roles. Their semantics depends on the verb and the verb usage in a sentence, or *verb sense*. The most frequent roles are A0 and A1 and, commonly, A0 stands for the *agent* and A1 corresponds to the *patient* or *theme* of the proposition. However, no consistent generalization can be made across different verbs or different senses of the same verb. PropBank takes the definition of verb senses from VerbNet, and for each verb and each sense defines the set of possible roles for that verb usage, called the *roleset*. The definition of rolesets is provided in the PropBank *Frames files*, which is made available for the shared task as an *official resource* to develop systems.

- **Adjuncts (AM-):** General arguments that any verb may take optionally. There are 13 types of adjuncts:

  AM-ADV : general-purpose
  AM-CAU : cause
  AM-DIR : direction
  AM-DIS : discourse marker
  AM-EXT : extent
  AM-LOC : location
  AM-MNR : manner
  AM-MOD : modal verb
  AM-NEG : negation marker
  AM-PNC : purpose
  AM-PRD : predication
  AM-REC : reciprocal
  AM-TMP : temporal

- **References (R-):** Arguments representing arguments realized in other parts of the sentence. The role of a reference is the same as the role of the referenced argument. The label is an R- tag prefixed to the label of the referent, e.g. R-A1.

- **Verbs (V):** Argument corresponding to the verb of the proposition. Each proposition has exaclty one verb argument.

We used PropBank-1.0. Most predicative verbs were annotated, although not all of them (for example, most of the occurrences of the verb "to have" and "to be" were not annotated). We applied procedures to check consistency of propositions, looking for overlapping arguments, and incorrect semantic role labels. Also, co-referenced arguments were annotated as a single item in PropBank, and we automatically distinguished between the referent and the reference with simple rules matching pronominal expressions, which were tagged as R arguments.

	Train.	Devel.	tWSJ	tBrown
Sentences	39,832	1,346	2,416	426
Tokens	950,028	32,853	56,684	7,159
Propositions	90,750	3,248	5,267	804
Verbs	3,101	860	982	351
Arguments	239,858	8,346	14,077	2,177
A0	61,440	2,081	3,563	566
A1	84,917	2,994	4,927	676
A2	19,926	673	1,110	147
A3	3,389	114	173	12
A4	2,703	65	102	15
A5	68	2	5	0
AA	14	1	0	0
AM	7	0	0	0
AM-ADV	8,210	279	506	143
AM-CAU	1,208	45	73	8
AM-DIR	1,144	36	85	53
AM-DIS	4,890	202	320	22
AM-EXT	628	28	32	5
AM-LOC	5,907	194	363	85
AM-MNR	6,358	242	344	110
AM-MOD	9,181	317	551	91
AM-NEG	3,225	104	230	50
AM-PNC	2,289	81	115	17
AM-PRD	66	3	5	1
AM-REC	14	0	2	0
AM-TMP	16,346	601	1,087	112
R-A0	4,112	146	224	25
R-A1	2,349	83	156	21
R-A2	291	5	16	0
R-A3	28	0	1	0
R-A4	7	0	1	0
R-AA	2	0	0	0
R-AM-ADV	5	0	2	0
R-AM-CAU	41	3	4	2
R-AM-DIR	1	0	0	0
R-AM-EXT	4	1	1	0
R-AM-LOC	214	9	21	4
R-AM-MNR	143	6	6	2
R-AM-PNC	12	0	0	0
R-AM-TMP	719	31	52	10

Table 1: Counts on the data sets.

A total number of 80 propositions were not compliant with our procedures (one in the Brown files, the rest in WSJ) and were filtered out from the CoNLL data sets.

Table 1 provides counts of the number of sentences, tokens, annotated propositions, distinct verbs, and arguments in the four data sets.

3.2 Preprocessing Systems

In this section we describe the selected processors that computed input annotations for the SRL systems. The annotations are: part-of-speech (PoS) tags, chunks, clauses, full syntactic trees and named entities. As it has been noted, participants were also

allowed to use any processor developed within the same WSJ partition.

The preprocessors correspond to the following state-of-the-art systems:

- UPC processors, consisting of:
  - PoS tagger: (Giménez and Màrquez, 2003), based on Support Vector Machines, and trained on WSJ sections 02-21.
  - Base Chunker and Clause Recognizer: (Carreras and Màrquez, 2003), based on Voted Perceptrons, trained on WSJ sections 02-21. These two processors form a coherent partial syntax of a sentence, that is, chunks and clauses form a partial syntactic tree.
- Full parser of Collins (1999), with "model 2". Predicts WSJ full parses, with information of the lexical head for each syntactic constituent. The PoS tags (required by the parser) have been computed with (Giménez and Màrquez, 2003).
- Full parser of Charniak (2000). Jointly predicts PoS tags and full parses.
- Named Entities predicted with the Maximum-Entropy based tagger of Chieu and Ng (2003). The tagger follows the CoNLL-2003 task setting (Tjong Kim Sang and De Meulder, 2003), and thus is not developed with WSJ data. However, we allowed its use because there is no available named entity recognizer developed with WSJ data. The reported performance on the CoNLL-2003 test is $F_1 = 88.31$, with Prec/Rec. at 88.12/88.51.

Tables 2 and 3 summarize the performance of the syntactic processors on the development and test sets. The performance of full parsers on the WSJ test is lower than that reported in the corresponding papers. The reason is that our evaluation figures have been computed in a strict manner with respect to punctuation tokens, while the full parsing community usually does not penalize for punctuation wrongly placed in the tree.[3] As it can be observed, the performance of all syntactic processors suffers a substantial loss in the Brown test set. Noticeably, the parser of Collins (1999) seems to be the more robust when moving from WSJ to Brown.

[3]Before evaluating Collins', we raised punctuation to the highest point in the tree, using a script that is available at the shared task webpage. Otherwise, the performance would have Prec./Recall figures below 37.

	Dev.	tWSJ	tBrown
UPC PoS-tagger	97.13	97.36	94.73
Charniak (2000)	92.01	92.29	87.89

Table 2: Accuracy (%) of PoS taggers.

4 A Review of Participant Systems

Nineteen systems participated in the CoNLL-2005 shared task. They approached the task in several ways, using different learning components and labeling strategies. The following subsections briefly summarize the most important properties of each system and provide a qualitative comparison between them, together with a quantitative evaluation on the development and test sets.

4.1 Learning techniques

Up to 8 different learning algorithms have been applied to train the learning components of participant systems. See the "ML-method" column of table 4 for a summary of the following information. Log–linear models and vector-based linear classifiers dominated over the rest. Probably, this is due to the versatility of the approaches and the availability of very good software toolkits.

In particular, 8 teams used the Maximum Entropy (ME) statistical framework (Che et al., 2005; Haghighi et al., 2005; Park and Rim, 2005; Tjong Kim Sang et al., 2005; Sutton and McCallum, 2005; Tsai et al., 2005; Yi and Palmer, 2005; Venkatapathy et al., 2005). Support Vector Machines (SVM) were used by 6 teams. Four of them with the standard polynomial kernels (Mitsumori et al., 2005; Tjong Kim Sang et al., 2005; Tsai et al., 2005; Pradhan et al., 2005b), another one using Gaussian kernels (Ozgencil and McCracken, 2005), and a last group using tree-based kernels specifically designed for the task (Moschitti et al., 2005). Another team used also a related learning approach, SNoW, which is a Winnow-based network of linear separators (Punyakanok et al., 2005).

Decision Tree learning (DT) was also represented

	Devel.			Test WSJ			Test Brown		
	P(%)	R(%)	F_1	P(%)	R(%)	F_1	P(%)	R(%)	F_1
UPC Chunker	94.66	93.17	93.91	95.26	94.52	94.89	92.64	90.85	91.73
UPC Clauser	90.38	84.73	87.46	90.93	85.94	88.36	84.21	74.32	78.95
Collins (1999)	85.02	83.55	84.28	85.63	85.20	85.41	82.68	81.33	82.00
Charniak (2000)	87.60	87.38	87.49	88.20	88.30	88.25	80.54	81.15	80.84

Table 3: Results of the syntactic parsers on the development, and WSJ and Brown test sets. Unlike in full parsing, the figures have been computed on a strict evaluation basis with respect to punctuation.

by Ponzetto and Strube (2005), who used C4.5. Ensembles of decision trees learned through the AdaBoost algorithm (AB) were applied by Màrquez et al. (2005) and Surdeanu and Turmo (2005). Tjong Kim Sang et al. (2005) applied, among others, Memory-Based Learning (MBL).

Regarding novel learning paradigms not applied in previous shared tasks, we find Relevant Vector Machine (RVM), which is a kernel–based linear discriminant inside the framework of Sparse Bayesian Learning (Johansson and Nugues, 2005) and Tree Conditional Random Fields (T-CRF) (Cohn and Blunsom, 2005), that extend the sequential CRF model to tree structures. Finally, Lin and Smith (2005) presented a proposal radically different from the rest, with very light learning components. Their approach (Consensus in Pattern Matching, CPM) contains some elements of Memory-based Learning and ensemble classification.

From the Machine Learning perspective, system combination is another interesting component observed in many of the proposals. This fact, which is a difference from last year shared task, is explained as an attempt of increasing the robustness and coverage of the systems, which are quite dependent on input parsing errors. The different outputs to combine are obtained by varying input information, changing learning algorithm, or considering n-best solution lists. The combination schemes presented include very simple voting-like combination heuristics, stacking of classifiers, and a global constraint satisfaction framework modeled with Integer Linear Programming. Global models trained to re-rank alternative outputs represent a very interesting alternative that has been proposed by two systems. All these issues are reviewed in detail in section 4.2.

4.2 SRL approaches

SRL is a complex task, which may be decomposed into a number of simpler decisions and annotating schemes in order to be addressed by learning techniques. Table 4 contains a summary of the main properties of the 19 systems presented. In this section we will explain the contents of that table by columns (from left-to-right).

One first issue to consider is the input structure to navigate in order to extract the constituents that will form labeled arguments. The majority of systems perform parse tree node labeling, searching for a one–to–one map between arguments and parse constituents. This information is summarized in the "synt" column of Table 4. "col", "cha", "upc" stand for the syntactic parse trees (the latter is partial) provided as input by the organization. Additionally, some teams used lists of n-best parsings generated by available tools ("n-cha" by Charniak parser; "n-bikel" by Bikel's implementation of Collins parser). Interestingly, Yi and Palmer (2005) retrained Ratnaparkhi's parser using the WSJ training sections enriched with semantic information coming from PropBank annotations. These are referred to as AN and AM parses. As it can be seen, Charniak parses were used by most of the systems. Collins parses were used also in some of the best performing systems based on combination.

The exceptions to the hierarchical processing are the systems by Pradhan et al. (2005b) and Mitsumori et al. (2005), which perform a chunking-based sequential tokenization. As for the former, the system is the same than the one presented in the 2004 edition. The system by Màrquez et al. (2005) explores hierarchical syntactic structures but selects, in a preprocess, a sequence of tokens to perform a sequential tagging afterwards.

	ML-method	synt	pre	label	embed	glob	post	comb	type
punyakanok	SNoW	n-cha,col	x&p	i+c	defer	yes	no	n-cha+col	ac-ILP
haghighi	ME	n-cha	?	i+c	dp-prob	yes	no	n-cha	re-rank
marquez	AB	cha,upc	seq	bio	!need	no	no	cha+upc	s-join
pradhan	SVM	cha,col/chunk	?	c/bio	?	no	no	cha+col→chunk	stack
surdeanu	AB	cha	prun	c	g-top	no	yes	no	–
tsai	ME,SVM	cha	x&p	c	defer	yes	no	ME+SVM	ac-ILP
che	ME	cha	no	c	g-score	no	yes	no	–
moschitti	SVM	cha	prun	i+c	!need	no	no	no	–
tjongkimsang	ME,SVM,TBL	cha	prun	i+c	!need	no	yes	ME+SVM+TBL	s-join
yi	ME	cha,AN,AM	x&p	i+c	defer	no	no	cha+AN+AM	ac-join
ozgencil	SVM	cha	prun	i+c	g-score	no	no	no	–
johansson	RVM	cha	softp	i+c	?	no	no	no	–
cohn	T-CRF	col	x&p	c	g-top	yes	no	no	–
park	ME	cha	prun	i+c	?	no	no	no	–
mitsumori	SVM	chunk	no	bio	!need	no	no	no	–
venkatapathy	ME	col	prun	i+c	frames	yes	no	no	–
ponzetto	DT	col	prun	c	g-top	no	yes	no	–
lin	CPM	cha	gt-para	i+c	!need	no	no	no	–
sutton	ME	n-bikel	x&p	i+c	dp-prob	yes	no	n-bikel	re-rank

Table 4: Main properties of the SRL strategies implemented by the participant teams, sorted by F_1 performance on the WSJ+Brown test set. **synt** stands for the syntactic structure explored; **pre** stands for pre-processing steps; **label** stands for the labeling strategy; **embed** stands for the technique to ensure non-embedding of arguments; **glob** stands for global optimization; **post** stands for post-processing; **comb** stands for system output combination, and **type** stands for the type of combination. Concrete values appearing in the table are explained in section 4.1. The symbol "?" stands for unknown values not reported by the system description papers.

In general, the presented systems addressed the SRL problem by applying different chained processes. In Table 4 the column "pre" summarizes preprocessing. In most of the cases this corresponds to a pruning procedure to filter out constituents that are not likely to be arguments. As in feature development, the related bibliography has been followed for pruning. For instance, many systems used the pruning strategy described in (Xue and Palmer, 2004) ("x&p") and other systems used the soft pruning rules described in (Pradhan et al., 2005a) ("softp"). Remarkably, Park and Rim (2005) parametrize the pruning procedure and then study the effect of being more or less aggressive at filtering constituents. In the case of Màrquez et al. (2005), pre-processing corresponds to a sequentialization of syntactic hierarchical structures. As a special case, Lin and Smith (2005) used the GT-PARA analyzer for converting parse trees into a flat representation of all predicates including argument boundaries.

The second stage, reflected in column "label" of Table 4, is the proper labeling of selected candidates. Most of the systems used a two-step procedure consisting of first identifying arguments (e.g., with a binary "null" vs. "non-null" classifier) and then classifying them. This is referred to as "i+c" in the table. Some systems address this phase in a single classification step by adding a "null" category to the multiclass problem (referred to as "c'). The methods performing a sequential tagging use a BIO tagging scheme ("bio"). As a special case, Moschitti et al. (2005) subdivide the "i+c" strategy into four phases: after identification, heuristics are applied to assure compatibility of identified arguments; and, before classifying arguments into roles, a preclassification into core vs. adjunct arguments is performed. Venkatapathy et al. (2005) use three labels instead of two in the identification phase : "null", "mandatory", and "optional".

Since arguments in a solution do not embed and most systems identify arguments as nodes in a hierarchical structure, non-embedding constraints must be resolved in order to generate a coherent argument labeling. The "embed" column of Table 4 accounts for this issue. The majority of systems applied specific greedy procedures that select a subset of consistent arguments. The families of heuristics to do that selection include prioritizing better scored

constituents ("g-score"), or selecting the arguments that are first reached in a top-down exploration ("g-top"). Some probabilistic systems include the non-embedding constraints within the dynamic programming inference component, and thus calculate the most probable coherent labeling ("dp-prob"). The "defer" value means that this is a combination system and that coherence of the individual system predictions is not forced, but deferred to the later combination step. As a particular case, Venkatapathy et al. (2005) use PropBank subcategorization frames to force a coherent solution. Note that tagging-based systems do not need to check non-embedding constraints ("!need" value).

The "glob" column of Table 4 accounts for the locality/globality of the process used to calculate the output solution given the argument prediction candidates. Systems with a "yes" value in that column define some kind of scoring function (possibly probabilistic) that applies to complete candidate solutions, and then calculate the solution that maximizes the scoring using an optimization algorithm.

Some systems use some kind of postprocessing to improve the final output of the system by correcting some systematic errors, or treating some types of simple adjunct arguments. This information is included in the "post" column of Table 4. In most of the cases, this postprocess is performed on the basis of simple ad-hoc rules. However, it is worth mentioning the work of Tjong Kim Sang et al. (2005) in which spelling error correction techniques are adapted for improving the resulting role labeling. In that system, postprocessing is applied before system combination.

Most of the best performing systems included a combination of different base subsystems to increase robustness of the approach and to gain coverage and independence from parse errors. Last 2 columns of Table 4 present this information. In the "comb" column the source of the combination is reported. Basically, the alternative outputs to combine can be generated by different input syntactic structures or n-best parse candidates, or by applying different learning algorithms to the same input information.

The type of combination is reported in the last column. Màrquez et al. (2005) and Tjong Kim Sang et al. (2005) performed a greedy merging of the arguments of base complete solutions ("s-join"). Yi and Palmer (2005) did also a greedy merging of arguments but taking into account not complete solutions but all candidate arguments labeled by base systems ("ac-join"). In a more sophisticated way, Punyakanok et al. (2005) and Tsai et al. (2005) performed global inference as constraint satisfaction using Integer Linear Programming, also taking into account all candidate arguments ("ac-ILP"). It is worth noting that the generalized inference applied in those papers allows to include, jointly with the combination of outputs, a number of linguistically-motivated constraints to obtain a coherent solution.

Pradhan et al. (2005b) followed a stacking approach by learning a chunk-based SRL system including as features the outputs of two syntax-based systems. Finally, Haghighi et al. (2005) and Sutton and McCallum (2005) performed a different approach by learning a re-ranking function as a global model on top of the base SRL models. Actually, Haghighi et al. (2005) performed a double selection step: an inner re-ranking of n-best solutions coming from the base system on a single tree; and an outer selection of the final solution among the candidate solutions coming from n-best parse trees. The re-ranking approach allows to define global complex features applying to complete candidate solutions to train the rankers.

4.3 Features

Looking at the description of the different systems, it becomes clear that the general type of features used in this edition is strongly based on previous work on the SRL task (Gildea and Jurafsky, 2002; Surdeanu et al., 2003; Pradhan et al., 2005a; Xue and Palmer, 2004). With no exception, all systems have made intensive use of syntax to extract features. While most systems work only on the output of a parser —Charniak's being the most preferred— some systems depend on many syntactic parsers. In the latter situation, either a system is a combination of many individual systems (each working with a different parser), or a system extracts features from many different parse trees while exploring the nodes of only one parse tree. Most systems have also considered named entities for extracting features.

The main types of features seen in this SRL edition can be divided into four general categories: (1) Features characterizing the structure of a candidate

	sources		argument							verb		arg–verb						p
	synt	ne	at	aw	ab	ac	ai	pp	sd	v	sc	rp	di	ps	pv	pi	sf	as
punyakanok	cha,col,upc	+	+	h	+	t	+	+	·	+	+	+	c	+	·	+	+	·
haghighi	cha	·	+	h	+	p,s	·	+	+	+	+	+	t	+	+	·	·	+
marquez	cha,upc	+	+	h	+	t	+	·	+	+	+	+	w,c	+	+	·	+	·
pradhan	cha,col,upc	+	+	h,c	+	p,s,t	+	+	·	+	+	+	c,t	+	+	+	+	·
surdeanu	cha	+	+	h,c	+	p,s	+	·	+	+	+	+	w,t	+	+	+	·	·
tsai	cha,upc	+	+	h	+	p,s,t	·	·	·	+	+	+	w	+	·	·	·	·
che	cha	+	+	h	+	·	·	+	·	+	+	+	t	+	+	·	·	·
moschitti	cha	·	+	h	+	p	+	+	·	+	+	+	t	+	+	·	+	·
tjongkimsang	cha	+	+	·	+	p,t	·	+	·	+	+	+	w,t	+	+	+	·	·
yi	cha,an,am	·	+	h,c	·	p,s	·	+	·	+	+	+	w	+	·	·	+	·
ozgencil	cha	·	+	h	·	p	·	·	·	+	+	+	·	+	+	·	·	·
johansson	cha,upc	+	+	h	·	·	·	·	·	+	+	+	·	+	+	·	·	·
cohn	col	·	+	h	+	p,s	·	+	·	+	+	+	w	+	·	+	+	·
park	cha	·	+	h,c	·	p	·	·	·	+	+	+	·	+	·	+	·	·
mitsumori	upc,cha	+	+	·	+	t	·	·	+	+	·	+	c,t	·	+	·	·	·
venkatapathy	col	+	+	h	+	·	·	·	·	+	·	+	·	+	·	·	·	·
ponzetto	col,upc	+	+	h	+	·	+	·	·	+	·	·	w,c,t	·	·	+	·	·
lin	cha	·	+	h	+	·	·	·	·	+	·	+	w	·	·	·	·	·
sutton	bik	·	+	h	+	p,s	·	·	·	+	·	+	·	+	·	·	·	+

Table 5: Main feature types used by the 19 participating systems in the CoNLL-2005 shared task, sorted by performance on the WSJ+Brown test set. **Sources: synt**: use of parsers, namely Charniak (cha), Collins (col), UPC partial parsers (upc), Bikel's Collins model (bik) and/or argument-enriched parsers (an,am); **ne**: use of named entities. **On the argument: at**: argument type; **aw**: argument words, namely the head (h) and/or content words (c); **ab**: argument boundaries, i.e. form and PoS of first and/or last argument words; **ac**: argument context, capturing features of the parent (p) and/or left/right siblings (s), or the tokens surrounding the argument (t); **ai**: indicators of the structure of the argument (e,g., on internal constituents, surrounding/boundary punctuation, governing category, etc.); **pp**: specific features for prepositional phrases; **sd**: semantic dictionaries. **On the verb: v**: standard verb features (voice, word/lemma, PoS); **sc**: subcategorization. **On the arg-verb relation: rp**: relative position; **di**: distance, based on words (w), chunks (c) or the syntactic tree (t); **ps**: standard path; **pv**: path variations; **pi**: scalar indicator variables on the path (of chunks, clauses, or other phrase types), common ancestor, etc.; **sf**: syntactic frame (Xue and Palmer, 2004); **On the complete proposition: as**: sequence of arguments of a proposition.

argument; (2) Features describing properties of the target verb predicate; (3) Features that capture the relation between the verb predicate and the constituent under consideration; and (4) Global features describing the complete argument labeling of a predicate. The rest of the section describes the most common feature types in each category. Table 5 summarizes the type of features exploited by systems.

To represent an argument itself, all systems make use of the syntactic type of the argument. Almost all teams used the heuristics of Collins (1999) to extract the head word of the argument, and used features that capture the form, lemma and PoS tag of the head. In the same line, some systems also use features of the content words of the argument, using the heuristics of Surdeanu et al. (2003). Very generally also, many systems extract features from the first and last words of the argument. Regarding the syntactic elements surrounding the argument, many systems working on full trees have considered the parent and siblings of the argument, capturing their syntactic type and head word. Differently, other systems have captured features from the left/right tokens surrounding the argument, which are typically words, but can be chunks or general phrases in systems that sequentialize the task (Màrquez et al., 2005; Pradhan et al., 2005b; Mitsumori et al., 2005). Many systems use a variety of indicator features that capture properties of the argument structure and its local syntactic annotations. For example, indicators of the immediate syntactic types that form the argument, flags raised by punctuation tokens in or nearby the argument, or the *governing category* feature of Gildea and Jurafsky (2002). It is also somewhat gen-

eral the use of specific features that apply when the constituent is a prepositional phrase, such as looking for the head word of the noun phrase within it. A few systems have also built semantic dictionaries from training data, that collect words appearing frequently in temporal, locative or other arguments.

To represent the predicate, all systems have used features codifying the form, lemma, PoS tag and voice of the verb. It is also of general use the subcategorization feature, capturing the syntactic rule that expands the parent of the predicate. Some systems captured statistics related to the frequency of a verb in training data (not in Table 5).

Regarding features related to an argument-verb pair, almost all systems use the simple feature describing the relative position between them. To a lesser degree, systems have computed distances from one to the other, based on the number of words or chunks between them, or based on the syntactic tree. Not surprisingly, all systems have extracted the path from the argument to the verb. While almost all systems use the standard path of (Gildea and Jurafsky, 2002), many have explored variations of it. A common one consists of the path from the argument to the lowest common ancestor of the verb and the argument. Another variation is the partial path, that is built of chunks and clauses only. Indicator features that capture scalar values of the path are also common, and concentrate mainly on looking at the common ancestor, capturing the difference of clausal levels, or looking for punctuation and other linguistic elements in the path. In this category, it is also noticeable the use of the *syntactic frame* feature, proposed by Xue and Palmer (2004).

Finally, in this edition two systems apply learning at a global context (Haghighi et al., 2005; Sutton and McCallum, 2005) and, consequently, they are able to extract features from a complete labeling of a predicate. Basically, the central feature in this context extracts the sequential pattern of predicate arguments. Then, this pattern can be enriched with syntactic categories, broken down into role-specific indicator variables, or conjoined with the predicate lemma.

Apart from basic feature extraction, combination of features has also been explored in this edition. Many of the combinations depart from the manually selected conjunctions of Xue and Palmer (2004).

4.4 Evaluation

A baseline rate was computed for the task. It was produced using a system developed in the past shared task edition by Erik Tjong Kim Sang, from the University of Amsterdam, The Netherlands. The baseline processor finds semantic roles based on the following seven rules:

- Tag target verb and successive particles as `V`.
- Tag `not` and `n't` in target verb chunk as `AM-NEG`.
- Tag modal verbs in target verb chunk as `AM-MOD`.
- Tag first `NP` before target verb as `A0`.
- Tag first `NP` after target verb as `A1`.
- Tag `that`, `which` and `who` before target verb as `R-A0`.
- Switch `A0` and `A1`, and `R-A0` and `R-A1` if the target verb is part of a passive `VP` chunk. A `VP` chunk is considered in passive voice if it contains a form of `to be` and the verb does not end in `ing`.

Table 6 presents the overall results obtained by the nineteen systems plus the baseline, on the development and test sets (i.e., Development, Test WSJ, Test Brown, and Test WSJ+Brown). The systems are sorted by the performance on the combined WSJ+Brown test set.

As it can be observed, all systems clearly outperformed the baseline. There are seven systems with a final F_1 performance in the 75-78 range, seven more with performances in the 70-75 range, and five with a performance between 65 and 70. The best performance was obtained by Punyakanok et al. (2005), which almost reached an F_1 at 80 in the WSJ test set and almost 78 in the combined test. Their results on the WSJ test equal the best results published so far on this task and datasets (Pradhan et al., 2005a), though they are not directly comparable due to a different setting in defining arguments not perfectly matching the predicted parse constituents. Since the evaluation in the shared task setting is more strict, we believe that the best results obtained in the shared task represent a new breakthrough in the SRL task.

It is also quite clear that the systems using combination are better than the individuals. It is worth noting that the first 4 systems are combined. The

	Development			Test WSJ			Test Brown			Test WSJ+Brown		
	P(%)	R(%)	F_1	P(%)	R(%)	F_1	P(%)	R(%)	F_1	P(%)	R(%)	F_1
punyakanok	80.05	74.83	77.35	82.28	76.78	79.44	73.38	62.93	67.75	81.18	74.92	77.92
haghighi	77.66	75.72	76.68	79.54	77.39	78.45	70.24	65.37	67.71	78.34	75.78	77.04
marquez	78.39	75.53	76.93	79.55	76.45	77.97	70.79	64.35	67.42	78.44	74.83	76.59
pradhan	80.90	75.38	78.04	81.97	73.27	77.37	73.73	61.51	67.07	80.93	71.69	76.03
surdeanu	79.14	71.57	75.17	80.32	72.95	76.46	72.41	59.67	65.42	79.35	71.17	75.04
tsai	81.13	72.42	76.53	82.77	70.90	76.38	73.21	59.49	65.64	81.55	69.37	74.97
che	79.65	71.34	75.27	80.48	72.79	76.44	71.13	59.99	65.09	79.30	71.08	74.97
moschitti	74.95	73.10	74.01	76.55	75.24	75.89	65.92	61.83	63.81	75.19	73.45	74.31
tjongkimsang	76.79	70.01	73.24	79.03	72.03	75.37	70.45	60.13	64.88	77.94	70.44	74.00
yi	75.70	69.99	72.73	77.51	72.97	75.17	67.88	59.03	63.14	76.31	71.10	73.61
ozgencil	73.57	71.87	72.71	74.66	74.21	74.44	65.52	62.93	64.20	73.48	72.70	73.09
johansson	73.40	70.85	72.10	75.46	73.18	74.30	65.17	60.59	62.79	74.13	71.50	72.79
cohn	73.51	68.98	71.17	75.81	70.58	73.10	67.63	60.08	63.63	74.76	69.17	71.86
park	72.68	69.16	70.87	74.69	70.78	72.68	64.58	60.31	62.38	73.35	69.37	71.31
mitsumori	71.68	64.93	68.14	74.15	68.25	71.08	63.24	54.20	58.37	72.77	66.37	69.43
venkatapathy	71.88	64.76	68.14	73.76	65.52	69.40	65.25	55.72	60.11	72.66	64.21	68.17
ponzetto	71.82	61.60	66.32	75.05	64.81	69.56	66.69	52.14	58.52	74.02	63.12	68.13
lin	70.11	61.96	65.78	71.49	64.67	67.91	65.75	52.82	58.58	70.80	63.09	66.72
sutton	64.43	63.11	63.76	68.57	64.99	66.73	62.91	54.85	58.60	67.86	63.63	65.68
baseline	50.00	28.98	36.70	51.13	29.16	37.14	62.66	33.07	43.30	52.58	29.69	37.95

Table 6: Overall precision, recall and F_1 rates obtained by the 19 participating systems in the CoNLL-2005 shared task on the development and test sets. Systems sorted by F_1 score on the WSJ+Brown test set.

best individual system on the task is that of Surdeanu and Turmo (2005), which obtained F_1=75.04 on the combined test set, about 3 points below than the best performing combined system. On the development set, that system achieved a performace of 75.17 (slightly below than the 75.27 reported by Che et al. (2005) on the same dataset). According to the description papers, we find that other individual systems, from which the combined systems are constructed, performed also very well. For instance, Tsai et al. (2005) report F_1=75.76 for a base system on the development set, Màrquez et al. (2005) report F_1=75.75, Punyakanok et al. (2005) report F_1=74.76, and Haghighi et al. (2005) report F_1=74.52.

The best results in the CoNLL-2005 shared task are 10 points better than those of last year edition. This increase in performance should be attributed to a combination of the following factors: 1) training sets have been substantially enlarged; 2) predicted parse trees are available as input information; and 3) more sophisticated combination schemes have been implemented. In order to have a more clear idea of the impact of enriching the syntactic information, we refer to (Màrquez et al., 2005), who developed an individual system based only on partial parsing ("upc" input information). That system performed F_1=73.57 on the development set, which is 2.18 points below the F_1=75.75 obtained by the same architecture using full parsing, and 4.47 points below the best performing combined system on the development set (Pradhan et al., 2005b).

Comparing the results across development and WSJ test corpora, we find that, with two exceptions, all systems experienced a significant increase in performance (normally between 1 and 2 F_1 points). This fact may be attributed to the different levels of difficulty found across WSJ sections. The linguistic processors and parsers perform slightly worse in the development set. As a consequence, the matching between parse nodes and actual arguments is lower.

Regarding the evaluation using the Brown test set, all systems experienced a severe drop in performance (about 10 F_1 points), even though the baseline on the Brown test set is higher than that of the WSJ test set. As already said in previous sections, all the linguistic processors, from PoS tagging to full parsing, showed a much lower performance than in the WSJ test set, evincing that their performance cannot be extrapolated across corpora. Presumably, this fact is the main responsible of the performace drop, though we do not discard an additional overfitting effect due to the design of specific features that do not generalize well. More im-

portantly, this results impose (again) a severe criticism on the current pipelined architecture for Natural Language Processing. Error propagation and amplification through the chained modules make the final output generalize very badly when changing the domain of application.

5 Conclusion

We have described the CoNLL-2005 shared task on semantic role labeling. Contrasting with the CoNLL-2004 edition, the current edition has incorporated the use of full syntax as input to the SRL systems, much larger training sets, and cross-corpora evaluation. The first two novelties have most likely contributed to an improvement of results. The latter has evinced a major drawback of natural language pipelined architectures.

Nineteen teams have participated to the task, contributing with a variety of learning algorithms, labeling strategies, feature design and experimentation. While, broadly, all systems make use of the same basic techniques described in existing SRL literature, some novel aspects have also been explored. A remarkable aspect, common in the four top-performing systems and many other, is that of combining many individual SRL systems, each working on different syntactic structures. Combining systems improves robustness, and overcomes the limitations in coverage that working with a single, non-correct syntactic structure imposes. The best system, presented by Punyakanok et al. (2005), achieves an F_1 at 79.44 on the WSJ test. This performance, of the same order than the best reported in literature, is still far from the desired behavior of a natural language analyzer. Furthermore, the performance of such SRL module in a real application will be about ten points lower, as demonstrated in the evaluation on the sentences from Brown.

We conclude with two open questions. First, what semantic knowledge is needed to improve the quality and performance of SRL systems. Second, beyond pipelines, what type of architectures and language learning methodology ensures a robust performance of processors.

Acknowledgements

Authors would like to thank the following people and institutions. The PropBank team, and specially Martha Palmer and Benjamin Snyder, for making available PropBank-1.0 and the prop-banked Brown files. The Linguistic Data Consortium, for issuing a free evaluation license for the shared task to use the TreeBank. Hai Leong Chieu and Hwee Tou Ng, for running their Named Entity tagger on the task data. Finally, the teams contributing to the shared task, for their great enthusiasm.

This work has been partially funded by the European Community (Chil - IP506909; PASCAL - IST-2002-506778) and the Spanish Ministry of Science and Technology (Aliado, TIC2002-04447-C02).

References

Xavier Carreras and Lluís Màrquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*, Borovets, Bulgaria.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2004*.

Eugene Charniak. 2000. A maximum-entropy inspired parser. In *Proceedings of NAACL-2000*.

Wanxiang Che, Ting Liu, Sheng Li, Yuxuan Hu, and Huaijun Liu. 2005. Semantic role labeling system using maximum entropy classifier. In *Proceedings of CoNLL-2005*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of CoNLL-2003*, Edmonton, Canada.

Trevor Cohn and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL-2005*.

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Charles J. Fillmore, Charles Wooters, and Collin F. Baker. 2001. Building a large lexical databank which provides deep semantics. In *Proceedings of the Pacific Asian Conference on Language, Informa tion and Computation*, Hong Kong, China.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Jesús Giménez and Lluís Màrquez. 2003. Fast and accurate part-of-speech tagging: The svm approach revisited. In *Proceedings of RANLP-2003*, Borovets, Bulgaria.

Aria Haghighi, Kristina Toutanova, and Christopher Manning. 2005. A joint model for semantic role labeling. In *Proceedings of CoNLL-2005*.

Richard Johansson and Pierre Nugues. 2005. Sparse bayesian classification of predicate arguments. In *Proceedings of CoNLL-2005*.

Chi-San Lin and Tony C. Smith. 2005. Semantic role labeling via consensus in pattern-matching. In *Proceedings of CoNLL-2005*.

Ken Litkowski. 2004. Senseval-3 task: Automatic labeling of semantic roles. In *Proceedings of the Senseval-3 ACL-SIGLEX Workshop*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19.

Lluís Màrquez, Pere Comas, Jesús Giménez, and Neus Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of CoNLL-2005*.

Tomohiro Mitsumori, Masaki Murata, Yasushi Fukuda, Kouichi Doi, and Hirohumi Doi. 2005. Semantic role labeling using support vector machines. In *Proceedings of CoNLL-2005*.

Alessandro Moschitti, Ana-Maria Giuglea, Bonaventura Coppola, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *Proceedings of CoNLL-2005*.

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics (ACL-2004)*.

Necati Ercan Ozgencil and Nancy McCracken. 2005. Semantic role labeling using libSVM. In *Proceedings of CoNLL-2005*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Kyung-Mi Park and Hae-Chang Rim. 2005. Maximum entropy based semantic role labeling. In *Proceedings of CoNLL-2005*.

Simone Paolo Ponzetto and Michael Strube. 2005. Semantic role labeling using lexical statistical information. In *Proceedings of CoNLL-2005*.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Daniel Jurafsky. 2005a. Support vector learning for semantic argument classification. *Machine Learning. Special issue on Speech and Natural Language Processing*. To appear.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005b. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL-2005*.

Vasin Punyakanok, Dan Roth, Wen-Tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Vasin Punyakanok, Peter Koomen, Dan Roth, and Wen tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of CoNLL-2005*.

Mihai Surdeanu and Jordi Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of CoNLL-2005*.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*, Sapporo, Japan.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*.

Erik Tjong Kim Sang, Sander Canisius, Antal van den Bosch, and Toine Bogers. 2005. Applying spelling error correction techniques for improving semantic role labelling. In *Proceedings of CoNLL-2005*.

Tzong-Han Tsai, Chia-Wei Wu, Yu-Chun Lin, and Wen-Lian Hsu. 2005. Exploiting full parsing information to label semantic roles using an ensemble of me and svm via integer linear programming. In *Proceedings of CoNLL-2005*.

Sriram Venkatapathy, Akshar Bharati, and Prashanth Reddy. 2005. Inferring semantic roles using subcategorization frames and maximum entropy model. In *Proceedings of CoNLL-2005*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Szu-ting Yi and Martha Palmer. 2005. The integration of syntactic parsing and semantic role labeling. In *Proceedings of CoNLL-2005*.

Inferring semantic roles using sub-categorization frames and maximum entropy model

Akshar Bharati, Sriram Venkatapathy and Prashanth Reddy
Language Technologies Research Centre, IIIT - Hyderabad, India.
{sriram,prashanth}@research.iiit.ac.in

Abstract

In this paper, we propose an approach for inferring semantic role using sub-categorization frames and maximum entropy model. Our approach aims to use the sub-categorization information of the verb to label the mandatory arguments of the verb in various possible ways. The ambiguity between the assignment of roles to mandatory arguments is resolved using the maximum entropy model. The unlabelled mandatory arguments and the optional arguments are labelled directly using the maximum entropy model such that their labels are not one among the frame elements of the sub-categorization frame used. Maximum entropy model is preferred because of its novel approach of smoothing. Using this approach, we obtained an F-measure of 68.14% on the development set of the data provided for the CONLL-2005 shared task. We show that this approach performs well in comparison to an approach which uses only the maximum entropy model.

1 Introduction

Semantic role labelling is the task of assigning appropriate semantic roles to the arguments of a verb. The semantic role information is important for various applications in NLP such as Machine Translation, Question Answering, Information Extraction etc. In general, semantic role information is useful for sentence understanding. We submitted our system for closed challenge at CONLL-2005 shared task. This task encourages participants to use novel machine learning techniques suited to the task of semantic role labelling. Previous approaches on semantic role labelling can be classified into three categories (1) Explicit Probabilistic methods (Gildea and Jurafsky, 2002). (2) General machine learning algorithms (Pradhan et al., 2003) (Lim et al., 2004) and (3) Generative model (Thompson et al., 2003).

Our approach has two stages; first, identification whether the argument is mandatory or optional and second, the classification or labelling of the arguments. In the first stage, the arguments of a verb are put into three classes, (1) mandatory, (2) optional or (3) null. Null stands for the fact that the constituent of the verb in the sentence is not an semantic argument of the verb. It is used to rule out the false argument of the verb which were obtained using the parser. The maximum entropy based classifier is used to classify the arguments into one of the above three labels.

After obtaining information about the nature of the non-null arguments, we proceed in the second stage to classify the mandatory and optional arguments into their semantic roles. The propbank sub-categorization frames are used to assign roles to the mandatory arguments. For example, in the sentence "John saw a tree", the sub-categorization frame "A0 v A1" would assign the roles A0 to John and A1 to tree respectively. After using all the sub-categorization frames of the verb irre-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 165–168, Ann Arbor, June 2005.

spective of the verb sense, there could be ambiguity in the assignment of semantic roles to mandatory arguments. The unlabelled mandatory arguments and the optional arguments are assigned the most probable semantic role which is not one of the frame elements of the sub-categorization frame using the maximum entropy model. Now, among all the sequences of roles assigned to the non-null arguments, the sequence which has the maximum joint probability is chosen. We obtained an accuracy of 68.14% using our approach. We also show that our approach performs better in comparision to an approach with uses a simple maximum entropy model. In section 4, we will talk about our approach in greater detail.

This paper is organised as follows, (2) Features, (3) Maximum entropy model, (4) Description of our system, (5) Results, (6) Comparison with our other experiments, (7) Conclusion and (8) Future work.

2 Features

The following are the features used to train the maximum entropy classifier for both the argument identification and argument classification. We used only simple features for these experiments, we are planning to use richer features in the near future.

1. Verb/Predicate.
2. Voice of the verb.
3. Constituent head and Part of Speech tag.
4. Label of the constituent.
5. Relative position of the constituent with respect to the verb.
6. The path of the constituent to the verb phrase.
7. Preposition of the constituent, NULL if it doesn't exist.

3 Maximum entropy model

The maximum entropy approach became the preferred approach of probabilistic model builders for its flexibility and its novel approach to smoothing (Ratnaparakhi, 1999).

Many classification tasks are most naturally handled by representing the instance to be classified as a vector of features. We represent features as binary functions of two arguments, f(a,H), where 'a' is the observation or the class and 'H' is the history. For example, a feature $f_i(a, H)$ is true if 'a' is Ram and 'H' is 'AGENT of a verb'. In a log linear model, the probability function $P(a|H)$ with a set of features $f_1, f_2, ...f_j$ that connects 'a' to the history 'H', takes the following form.

$$P(a|H) = \frac{e^{\sum_i \lambda_i(a,H) * f_i(a,H)}}{Z(H)}$$

Here λ_i's are weights between negative and positive infinity that indicate the relative importance of a feature: the more relevant the feature to the value of the probability, the higher the absolute value of the associated lambda. Z(H), called the partition function, is the normalizing constant (for a fixed H).

4 Description of our system

Our approach labels the semantic roles in two stages, (1) argument identification and (2) argument classification. As input to our system, we use full syntactic information (Collins, 1999), Named-entities, Verb senses and Propbank frames. For our experiments, we use Zhang Le's Maxent Toolkit [1], and the L-BFGS parameter estimation algorithm with Gaussian prior smoothing (Chen and Rosenfield, 1999).

4.1 Argument identification

The first task in this stage is to find the candidate arguments and their boundaries using a parser. We use Collins parser to infer a list of candidate arguments for every predicate. The following are some of the sub-stages in this task.

- Convert the CFG tree given by Collins parser to a dependency tree.
- Eliminate auxilliary verbs etc.
- Mark the head of relative clause as an argument of the verb.

[1] http://www.nlplab.cn/zhangle/maxent_toolkit.html

- If a verb is modified by another verb, the syntactic arguments of the superior verb are considered as shared arguments between both the verbs.

- If a prepositional phrase attached to a verb contains more than one noun phrase, attach the second noun phrase to the verb.

The second task is to filter out the constituents which are not really the arguments of the predicate. Given our approach towards argument classification, we also need information about whether an argument is mandatory or optional. Hence, in this stage the constituents are marked using three labels, (1) MANDATORY argument, (2) OPTIONAL argument and (3) NULL, using a maximum entropy classifier. For example, a sentence "John was playing football in the evening", "John" is marked MANDATORY, "football" is marked MANDATORY and "in the evening" is marked OPTIONAL.

For training, the Collins parser is run on the training data and the syntactic arguments are identified. Among these arguments, the ones which do not exist in the propbank annotation of the training data are marked as null. Among the remaining arguments, the arguments are marked as mandatory or optional according to the propbank frame information. Mandatory roles are those appearing in the propbank frames of the verb and its sense, the rest are marked as optional. A propbank frame contains information as illustrated by the following example:

If Verb = play, sense = 01,
then the roles A0, A1 are MANDATORY.

4.2 Argument classification

Argument classification is done in two steps. In the first step, the propbank sub-categorization frames are used to assign the semantic roles to the mandatory arguments in the order specified by the sub-categorization frames. Sometimes, the number of mandatory arguments of a verb in the sentence may be less than the number of roles which can be assigned by the sub-categorization frame. For example, in the sentence

"MAN1 MAN2 V MAN3 OPT1", roles could be assigned in the following two possible ways by the sub-categorization frame "A0 v A1" of verb V1.

- A0[MAN1] MAN2 V1 A1[MAN3] OPT1

- MAN1 A0[MAN2] V A1[MAN3] OPT1

In the second step, the task is to label the unlabelled mandatory arguments and the arguments which are marked as optional. This is done by marking these arguments with the most probable semantic role which is not one of the frame elements of the sub-categorization frame "A0 v A1". In the above example, the unlabelled mandatory arguments and the optional arguments cannot be labelled as either A0 or A1. Hence, after this step, the following might be the role-labelling for the sentence "MAN1 MAN2 V1 MAN3 OPT1".

- A0[MAN1] AM-TMP[MAN2] V1 A1[MAN3] AM-LOC[OPT1]

- AM-MNC[MAN1] A0[MAN2] V1 A1[MAN3] AM-LOC[OPT1]

The best possible sequence of semantic roles ($\bar{R}$) is decided by the taking the product of probabilities of individual assignments. This also disambiguates the ambiguity in the assignment of mandatory roles. The individual probabilities are computed using the maximum entropy model. For a sequence $\vec{R}$, the product of the probabilities is defined as

$$P(\vec{R}) = \Pi_{R_i \in \vec{R}} P(R_i | Arg_i)$$

The best sequence of semantic roles $\bar{R}$ is defined as

$$\bar{R} = argmax\ P(\vec{R})$$

For training the maximum entropy model, the outcomes are all the possible semantic roles. The list of sub-categorization frames for a verb is obtained from the training data using information about mandatory roles from the propbank. The propbank sub-categorization frames are also appended to this list.

We present our results in the next section.

	Precision	Recall	$F_{\beta=1}$
Development	71.88%	64.76%	68.14
Test WSJ	73.76%	65.52%	69.40
Test Brown	65.25%	55.72%	60.11
Test WSJ+Brown	72.66%	64.21%	68.17

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	73.76%	65.52%	69.40
A0	85.17%	73.34%	78.81
A1	74.08%	66.08%	69.86
A2	54.51%	48.47%	51.31
A3	52.54%	35.84%	42.61
A4	71.13%	67.65%	69.35
A5	25.00%	20.00%	22.22
AM-ADV	52.18%	47.23%	49.59
AM-CAU	60.42%	39.73%	47.93
AM-DIR	45.65%	24.71%	32.06
AM-DIS	75.24%	73.12%	74.17
AM-EXT	73.68%	43.75%	54.90
AM-LOC	50.80%	43.53%	46.88
AM-MNR	47.24%	49.71%	48.44
AM-MOD	93.67%	91.29%	92.46
AM-NEG	94.67%	92.61%	93.63
AM-PNC	42.02%	43.48%	42.74
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	74.13%	66.97%	70.37
R-A0	82.27%	80.80%	81.53
R-A1	73.28%	61.54%	66.90
R-A2	75.00%	37.50%	50.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	100.00%	57.14%	72.73
R-AM-MNR	25.00%	16.67%	20.00
R-AM-TMP	70.00%	53.85%	60.87
V	97.28%	97.28%	97.28

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

5 Results

The results of our approach are presented in table 1.

When we used an approach which uses a simple maximum entropy model, we obtained an F-measure of 67.03%. Hence, we show that the sub-categorization frames help in predicting the semantic roles of the mandatory arguments, thus improving the overall performance.

6 Conclusion

In this paper, we propose an approach for inferring semantic role using sub-categorization frames and maximum entropy model. Using this approach, we obtained an F-measure of 68.14% on the development set of the data provided for the CONLL-2005 shared task.

7 Future work

We have observed that the main limitation of our system was in argument identification. Currently, the recall of the arguments inferred from the output of the parser is 75.52% which makes it the upper bound of recall of our system. In near future, we would focus on increasing the upper bound of recall. In this direction, we would also use the partial syntactic information. The accuracy of the first stage of our approach would increase if we include the mandatory/optional information for training the parser (Yi and Palmer, 1999).

8 Acknowledgements

We would like to thank Prof. Rajeev Sangal, Dr. Sushama Bendre and Dr. Dipti Misra Sharma for guiding us in this project. We would like to thank Szu-ting for giving some valuable advice.

References

S. Chen and R. Rosenfield. 1999. A gaussian prior for smoothing maximum entropy models.

M. Collins. 1999. Head driven statistical models for natural language processing.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles.

Hwang Young Sook Lim, Joon-H and, So-Young Park, and Hae-Chang Rim. 2004. Semantic role labelling using maximum entropy model.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James. H. Martin, and Daniel Jurafsky. 2003. *Support Vector Learning for Semantic Argument Classification*.

Adwait Ratnaparakhi. 1999. Learning to parse natural language with maximum entropy models.

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labelling.

Szu-ting Yi and M. Palmer. 1999. The integration of syntactic parsing and semantic role labeling.

Semantic Role Labelling with Tree Conditional Random Fields

Trevor Cohn and **Philip Blunsom**
University of Melbourne, Australia
tacohn@csse.unimelb.edu.au and pcbl@csse.unimelb.ed.au

Abstract

In this paper we apply conditional random fields (CRFs) to the semantic role labelling task. We define a random field over the structure of each sentence's syntactic parse tree. For each node of the tree, the model must predict a semantic role label, which is interpreted as the labelling for the corresponding syntactic constituent. We show how modelling the task as a tree labelling problem allows for the use of efficient CRF inference algorithms, while also increasing generalisation performance when compared to the equivalent maximum entropy classifier. We have participated in the CoNLL-2005 shared task closed challenge with full syntactic information.

1 Introduction

The semantic role labelling task (SRL) involves identifying which groups of words act as arguments to a given predicate. These arguments must be labelled with their role with respect to the predicate, indicating how the proposition should be semantically interpreted.

We apply conditional random fields (CRFs) to the task of SRL proposed by the CoNLL shared task 2005 (Carreras and Màrquez, 2005). CRFs are undirected graphical models which define a conditional distribution over labellings given an observation (Lafferty et al., 2001). These models allow for the use of very large sets of arbitrary, overlapping and non-independent features. CRFs have been applied with impressive empirical results to the tasks of named entity recognition (McCallum and Li, 2003; Cohn et al., 2005), part-of-speech (PoS) tagging (Lafferty et al., 2001), noun phrase chunking (Sha and Pereira, 2003) and extraction of table data (Pinto et al., 2003), among other tasks.

While CRFs have not been used to date for SRL, their close cousin, the maximum entropy model has been, with strong generalisation performance (Xue and Palmer, 2004; Lim et al., 2004). Most CRF implementations have been specialised to work with chain structures, where the labels and observations form a linear sequence. Framing SRL as a linear tagging task is awkward, as there is no easy model of adjacency between the candidate constituent phrases.

Our approach simultaneously performs both constituent selection and labelling, by defining an undirected random field over the parse tree. This allows the modelling of interactions between parent and child constituents, and the prediction of an optimal argument labelling for all constituents in one pass. The parse tree forms an acyclic graph, meaning that efficient exact inference in a CRF is possible using belief propagation.

2 Data

The data used for this task was taken from the Propbank corpus, which supplements the Penn Treebank with semantic role annotation. Full details of the data set are provided in Carreras and Màrquez (2005).

2.1 Data Representation

From each training instance we derived a tree, using the parse structure from the Collins parser. The

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 169–172, Ann Arbor, June 2005.

nodes in the trees were relabelled with a semantic role label indicating how their corresponding syntactic constituent relates to each predicate, as shown in Figure 1. The role labels are shown as subscripts in the figure, and both the syntactic categories and the words at the leaves are shown for clarity only – these were not included in the tree. Additionally, the dashed lines show those edges which were pruned, following Xue and Palmer (2004) – only nodes which are siblings to a node on the path from the verb to the root are included in the tree. Child nodes of included prepositional phrase nodes are also included. This reduces the size of the resultant tree whilst only very occasionally excluding nodes which should be labelled as an argument.

The tree nodes were labelled such that only argument constituents received the argument label while all argument children were labelled as outside, `O`. Where there were parse errors, such that no constituent exactly covered the token span of an argument, the smaller subsumed constituents were all given the argument label.

We experimented with two alternative labelling strategies: labelling a constituent's children with a new 'inside' label, and labelling the children with the parent's argument label. In the figure, the IN and NP children of the PP would be affected by these changes, both receiving either the inside `I` label or `AM-LOC` label under the respective strategies. The inside strategy performed nearly identically to the standard (outside) strategy, indicating that either the model cannot reliably predict the inside argument, or that knowing that the children of a given node are inside an argument is not particularly useful in predicting its label. The second (duplication) strategy performed extremely poorly. While this allowed the internal argument nodes to influence their ancestor towards a particular labelling, it also dramatically increased the number of nodes given an argument label. This lead to spurious over-prediction of arguments.

The model is used for decoding by predicting the maximum probability argument label assignment to each of the unlabelled trees. When these predictions were inconsistent, and one argument subsumed another, the node closest to the root of the tree was deemed to take precedence over its descendants.

3 Model

We define a CRF over the labelling $\mathbf{y}$ given the observation tree $\mathbf{x}$ as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \sum_{c \in \mathcal{C}} \sum_{k} \lambda_k f_k(c, \mathbf{y}_c, \mathbf{x})$$

where $\mathcal{C}$ is the set of cliques in the observation tree, λ_k are the model's parameters and $f_k(\cdot)$ is the feature function which maps a clique labelling to a vector of scalar values. The function $Z(\cdot)$ is the normalising function, which ensures that p is a valid probability distribution. This can be restated as:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \exp \left\{ \sum_{v \in \mathcal{C}_1} \sum_{k} \lambda_k g_k(v, \mathbf{y}_v, \mathbf{x}) \right. \\ &\quad \left. + \sum_{u,v \in \mathcal{C}_2} \sum_{j} \lambda_j h_j(u, v, \mathbf{y}_u, \mathbf{y}_v, \mathbf{x}) \right\} \end{aligned}$$

where $\mathcal{C}_1$ are the vertices in the graph and $\mathcal{C}_2$ are the maximal cliques in the graph, consisting of all *(parent, child)* pairs. The feature function has been split into g and h, each dealing with one and two node cliques respectively.

Preliminary experimentation without any pair-wise features (h), was used to mimic a simple maximum entropy classifier. This model performed considerably worse than the model with the pair-wise features, indicating that the added complexity of modelling the parent-child interactions provides for more accurate modelling of the data.

The log-likelihood of the training sample was optimised using limited memory variable metric (LMVM), a gradient based technique. This required the repeated calculation of the log-likelihood and its derivative, which in turn required the use of dynamic programming to calculate the marginal probability of each possible labelling of every clique using the sum-product algorithm (Pearl, 1988).

4 Features

As the conditional random field is conditioned on the observation, it allows feature functions to be defined over any part of the observation. The tree structure requires that features incorporate either a node labelling or the labelling of a parent and its

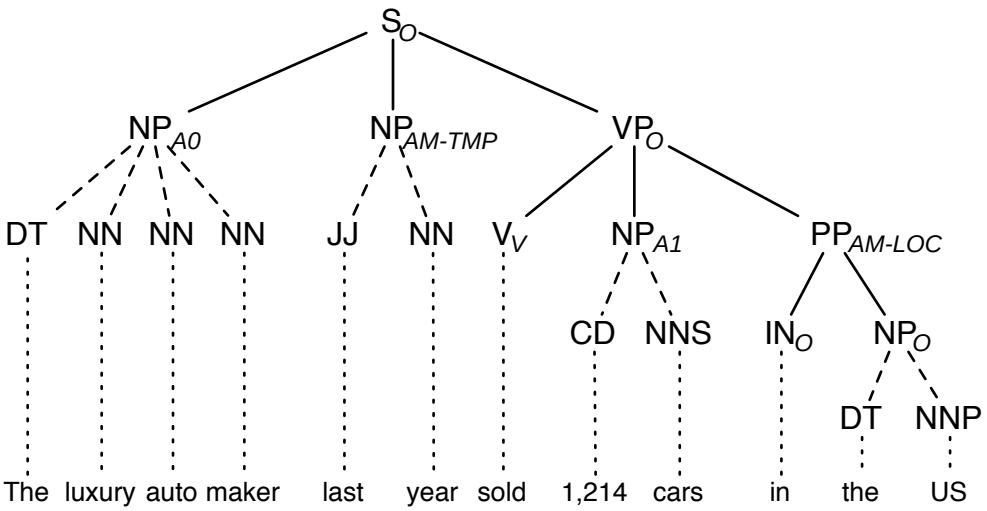


Figure 1: Syntax tree labelled for semantic roles with respect to the predicate *sell*. The subscripts show the role labels, and the dotted and dashed edges are those which are pruned from the tree.

child. We have defined node and pairwise clique features using data local to the corresponding syntactic node(s), as well as some features on the predicate itself.

Each feature type has been made into binary feature functions g and h by combining *(feature type, value)* pairs with a label, or label pair, where this combination was seen at least once in the training data. The following feature types were employed, most of which were inspired by previous works:

Basic features: {*Head word, head PoS, phrase syntactic category, phrase path, position relative to the predicate, surface distance to the predicate, predicate lemma, predicate token, predicate voice, predicate sub-categorisation, syntactic frame*}. These features are common to many SRL systems and are described in Xue and Palmer (2004).

Context features {*Head word of first NP in preposition phrase, left and right sibling head words and syntactic categories, first and last word in phrase yield and their PoS, parent syntactic category and head word*}. These features are described in Pradhan et al. (2005).

Common ancestor of the verb The syntactic category of the deepest shared ancestor of both the verb and node.

Feature conjunctions The following features were conjoined: { *predicate lemma + syntactic category, predicate lemma + relative position, syntactic category + first word of the phrase*}.

Default feature This feature is always on, which allows the classifier to model the prior probability distribution over the possible argument labels.

Joint features These features were only defined over pair-wise cliques: {*whether the parent and child head words do not match, parent syntactic category + and child syntactic category, parent relative position + child relative position, parent relative position + child relative position + predicate PoS + predicate lemma*}.

5 Experimental Results

The model was trained on the full training set after removing unparsable sentences, yielding 90,388 predicates and 1,971,985 binary features. A Gaussian prior was used to regularise the model, with variance $\sigma^2 = 1$. Training was performed on a 20 node PowerPC cluster, consuming a total of 62Gb of RAM and taking approximately 15 hours. Decoding required only 3Gb of RAM and about 5 minutes for the 3,228 predicates in the development set. Results are shown in Table 1.

	Precision	Recall	$F_{\beta=1}$
Development	73.51%	68.98%	71.17
Test WSJ	75.81%	70.58%	73.10
Test Brown	67.63%	60.08%	63.63
Test WSJ+Brown	74.76%	69.17%	71.86

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	75.81%	70.58%	73.10
A0	82.21%	79.48%	80.82
A1	74.56%	71.26%	72.87
A2	63.93%	56.85%	60.18
A3	63.95%	54.34%	58.75
A4	68.69%	66.67%	67.66
A5	0.00%	0.00%	0.00
AM-ADV	54.73%	48.02%	51.16
AM-CAU	75.61%	42.47%	54.39
AM-DIR	54.17%	30.59%	39.10
AM-DIS	77.74%	73.12%	75.36
AM-EXT	65.00%	40.62%	50.00
AM-LOC	60.67%	54.82%	57.60
AM-MNR	54.66%	49.42%	51.91
AM-MOD	98.34%	96.55%	97.44
AM-NEG	99.10%	96.09%	97.57
AM-PNC	49.47%	40.87%	44.76
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	77.20%	68.54%	72.61
R-A0	87.78%	86.61%	87.19
R-A1	82.39%	75.00%	78.52
R-A2	0.00%	0.00%	0.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	71.05%	51.92%	60.00
V	98.73%	98.63%	98.68

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

6 Conclusion

Conditional random fields proved useful in modelling the semantic structure of text when provided with a parse tree. Our novel use of a tree structure derived from the syntactic parse, allowed for parent-child interactions to be accurately modelled, which provided an improvement over a standard maximum entropy classifier. In addition, the parse constituent structure proved quite appropriate to the task, more so than modelling the data as a sequence of words or chunks, as has been done in previous approaches.

Acknowledgements

We would both like to thank our research supervisor Steven Bird for his comments and feedback on this work. The research undertaken for this paper was supported by an Australian Postgraduate Award scholarship, a Melbourne Research Scholarship and a Melbourne University Postgraduate Overseas Research Experience Scholarship.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the CoNLL-2005*.

Trevor Cohn, Andrew Smith, and Miles Osborne. 2005. Scaling conditional random fields using error correcting codes. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. To appear.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289.

Joon-Ho Lim, Young-Sook Hwang, So-Young Park, and Hae-Chang Rim. 2004. Semantic role labeling using maximum entropy model. In *Proceedings of the CoNLL-2004 Shared Task*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Natural Language Learning*, pages 188–191.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

David Pinto, Andrew McCallum, Xing Wei, and Bruce Croft. 2003. Table extraction using conditional random fields. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–242.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. In *To appear in Machine Learning journal, Special issue on Speech and Natural Language Processing*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, pages 213–220.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

A Joint Model for Semantic Role Labeling

Aria Haghighi
Dept of Computer Science
Stanford University
Stanford, CA, 94305
aria42@stanford.edu

Kristina Toutanova
Dept of Computer Science
Stanford University
Stanford, CA, 94305
kristina@cs.stanford.edu

Christopher D. Manning
Dept of Computer Science
Stanford University
Stanford, CA, 94305
manning@cs.stanford.edu

Abstract

We present a semantic role labeling system submitted to the closed track of the CoNLL-2005 shared task. The system, introduced in (Toutanova et al., 2005), implements a joint model that captures dependencies among arguments of a predicate using log-linear models in a discriminative re-ranking framework. We also describe experiments aimed at increasing the robustness of the system in the presence of syntactic parse errors. Our final system achieves F1-Measures of 76.68 and 78.45 on the development and the WSJ portion of the test set, respectively.

1 Introduction

It is evident that there are strong statistical patterns in the syntactic realization and ordering of the arguments of verbs; for instance, if an active predicate has an A0 argument it is very likely to come before an A1 argument. Our model aims to capture such dependencies among the labels of nodes in a syntactic parse tree.

However, building such a model is computationally expensive. Since the space of possible joint labelings is exponential in the number of parse tree nodes, a model cannot exhaustively consider these labelings unless it makes strong independence assumptions. To overcome this problem, we adopt a discriminative re-ranking approach reminiscent of (Collins, 2000). We use a local model, which labels arguments independently, to generate a smaller number of likely joint labelings. These candidate labelings are in turn input to a joint model which can use global features and re-score the candidates. Both the local and global re-ranking models are log-linear (maximum entropy) models.

In the following sections, we briefly describe our local and joint models and the system architecture for combining them. We list the features used by our models, with an emphasis on new features, and compare the performance of a local and a joint model on the CoNLL shared task. We also study an approach to increasing the robustness of the semantic role labeling system to syntactic parser errors, by considering multiple parse trees generated by a statistical parser.

2 Local Models

Our local model labels nodes in a parse tree independently. We decompose the probability over labels (all argument labels plus NONE), into a product of the probability over ARG and NONE, and a probability over argument labels given that a node is an ARG. This can be seen as chaining an *identification* and a *classification* model. The identification model classifies each phrase as either an argument or non-argument and our classification model labels each potential argument with a specific argument label. The two models use the same features.

Previous research (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Carreras and Màrquez, 2004) has identified many useful features for local identification and classification. Below we list the features and hand-picked conjunctions of features used in our local models. The ones denoted with asterisks (*) were not present in (Toutanova et al., 2005). Although most of these features have been described in previous work, some features, described in the next section, are – to our knowledge – novel.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 173–176, Ann Arbor, June 2005.

- **Phrase-Type** Syntactic category of node
- **Predicate Lemma** Stemmed target verb
- **Path** Sequence of phrase types between the predicate and node, with ↑, ↓ to indicate direction
- **Position** Before or after predicate
- **Voice** Voice of predicate
- **Head-Word of Phrase**
- **Head-POS** POS tag of head word
- **Sub-Cat** CFG expansion of predicate's parent
- **First/Last Word**
- **Left/Right Sister Phrase-Type**
- **Left/Right Sister Head-Word/Head-POS**
- **Parent Phrase-Type**
- **Parent POS/Head-Word**
- **Ordinal Tree Distance** Phrase-type concatenated with the length of the **Path** feature
- **Node-LCA Partial Path** Path from the node to the lowest common ancestor of the predicate and the node
- **PP Parent Head-Word** If the parent of the node is a PP, the parent's head-word
- **PP NP Head-Word/Head-POS** For a PP, retrieve the head-word /head-POS of its rightmost NP
- **Temporal Keywords*** Is the head of the node a temporal word e.g 'February' or 'afternoon'
- **Missing subject*** Is the predicate missing a subject in the"standard" location
- **Projected path*** Path from the maximal extended projection of the predicate to the node
- **Predicate Lemma & Path**
- **Predicate Lemma & Head-Word**
- **Predicate Lemma & Phrase-Type**
- **Voice & Position**
- **Predicate Lemma & PP Parent Head-Word**
- **Path & Missing subject***
- **Projected path & Missing subject***

2.1 Additional Local Features

We found that a large source of errors for A0 and A1 stemmed from cases such as those illustrated in Figure 1, where arguments were dislocated by raising or controlling verbs. Here, the predicate, *expected*, does not have a subject in the typical position – indicated by the empty NP – since the auxiliary *is* has raised the subject to its current position. In order to capture this class of examples, we use a binary feature, **Missing Subject**, indicating whether the predicate is "missing" its subject, and use this feature in conjunction with the **Path** feature, so that we learn typical paths to raised subjects conditioned on the absence of the subject in its typical position.

In the particular case of Figure 1, there is another instance of an argument being quite far from its predicate. The predicate *widen* shares *the trade gap* with *expect* as a A1 argument. However, as *expect* is a raising verb, *widen*'s subject is not in its typical position either, and we should expect to find it in the same positions as *expected*'s subject. This indicates it may be useful to use the path relative to *expected* to find arguments for *widen*. In general, to identify certain arguments of predicates embedded in auxiliary and infinitival VPs we expect it to be helpful to take the path from the maximum extended projection of the predicate – the highest VP in the chain of VP's dominating the predicate. We introduce a new path feature, **Projected Path**, which takes the path from the maximal extended projection to an argument node. This feature applies only when the argument is not dominated by the maximal projection, (e.g., direct objects). These features also handle other cases of discontinuous and non-local dependencies, such as those arising due to controller verbs. For a local model, these new features and their conjunctions improved F1-Measure from 73.80 to 74.52 on the development set. Notably, the F1-Measure of A0 increased from 81.02 to 83.08.

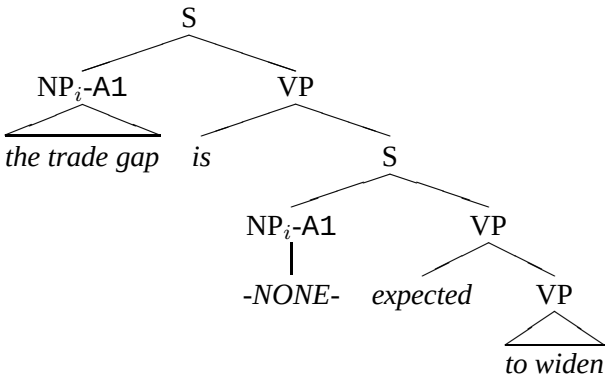


Figure 1: Example of displaced arguments

3 Joint Model

Our joint model, in contrast to the local model, collectively scores a labeling of all nodes in the parse tree. The model is trained to re-rank a set of N likely labelings according to the local model. We find the exact top N consistent[1] most likely local model labelings using a simple dynamic program described in (Toutanova et al., 2005).

[1] A labeling is consistent if satisfi es the constraint that argument phrases do not overlap.

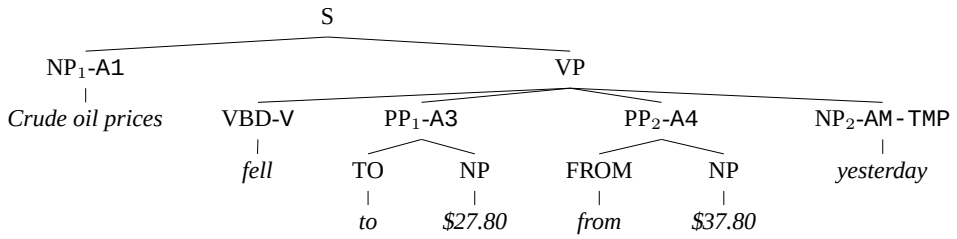


Figure 2: An example tree with semantic role annotations.

Most of the features we use are described in more detail in (Toutanova et al., 2005). Here we briefly describe these features and introduce several new joint features (denoted by *). A labeling L of all nodes in the parse tree specifies a candidate argument frame – the sequence of all nodes labeled with a non-NONE label according to L. The joint model features operate on candidate argument frames, and look at the labels and internal features of the candidate arguments. We introduce them in the context of the example in Figure 2. The candidate argument frame corresponding to the correct labeling for the tree is: [NP$_1$-A1,VBD-V,PP$_1$-A3,PP$_2$-A4,NP$_2$-AM-TMP].

- **Core arguments label sequence:** The sequence of labels of core arguments concatenated with the predicate voice. Example: [voice:active: A1,V,A3,A4] A back-off feature which substitutes specific argument labels with a generic argument (A) label is also included.
- **Flattened core arguments label sequence*:** Same as the previous but merging consecutive equal labels.
- **Core arguments label and annotated phrase type sequence:** The sequence of labels of core arguments together with annotated phrase types. Phrase types are annotated with the head word for PP nodes, and with the head POS tag for S and VP nodes. Example: [voice:active: NP-A1,V,PP-to-A3,PP-from-A4]. A back-off to generic A labels is also included. Also a variant that adds the predicate stem.
- **Repeated core argument labels with phrase types:** Annotated phrase types for nodes with the same core argument label. This feature captures, for example, the tendency of WHNP referring phrases to occur as the second phrase having the same label as a preceding NP phrase.
- **Repeated core argument labels with phrase types and sister/adjacency information*:** Similar to the previous feature, but also indicates whether all repeated arguments are sisters in the parse tree, or whether all repeated arguments are adjacent in terms of word spans. These features can provide robustness to parser errors, making it more likely to label adjacent phrases incorrectly split by the parser with the same label.

4 Combining Local and Joint Models

It is useful to combine the joint model score with a local model score, because the local model has been trained using all negative examples, whereas the joint model has been trained only on likely argument frames . Our final score is given by a mixture of the local and joint model's log-probabilities: $score_{SRL}(L|t) = \alpha\, score_\ell(L|t) + score_J(L|t)$, where $score_\ell(L|t)$ is the local score of L, $score_J(L|t)$ is the corresponding joint score, and α is a tunable parameter. We search among the top N candidate labelings proposed by the local model, for the labeling that maximizes the final score.

5 Increasing Robustness to Parser Errors

It is apparent that role labeling is very sensitive to the correctness of the given parse tree. If an argument does not correspond to a constituent in a parse tree, our model will not be able to consider the correct phrase.

One way to address this problem is to utilize alternative parses. Recent releases of the Charniak parser (Charniak, 2000) have included an option to provide the top k parses of a given sentence according to the probability model of the parser. We use these alternative parses as follow: Suppose $t_1, \ldots, t_k$ are trees for sentence s with given probabilities $P(t_i|s)$ by the parser. Then for a fixed predicate v, let L_i

	Precision	Recall	$F_{\beta=1}$
Development	77.66%	75.72%	76.68
Test WSJ	79.54%	77.39%	78.45
Test Brown	70.24%	65.37%	67.71
Test WSJ+Brown	78.34%	75.78%	77.04

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	79.54%	77.39%	78.45
A0	88.32%	88.30%	88.31
A1	78.61%	78.40%	78.51
A2	72.55%	68.11%	70.26
A3	73.08%	54.91%	62.71
A4	77.42%	70.59%	73.85
A5	100.00%	80.00%	88.89
AM-ADV	58.20%	51.19%	54.47
AM-CAU	63.93%	53.42%	58.21
AM-DIR	52.56%	48.24%	50.31
AM-DIS	76.56%	80.62%	78.54
AM-EXT	73.68%	43.75%	54.90
AM-LOC	61.52%	55.92%	58.59
AM-MNR	58.33%	56.98%	57.65
AM-MOD	97.85%	99.09%	98.47
AM-NEG	97.41%	98.26%	97.84
AM-PNC	49.50%	43.48%	46.30
AM-PRD	100.00%	20.00%	33.33
AM-REC	0.00%	0.00%	0.00
AM-TMP	74.85%	67.34%	70.90
R-A0	92.63%	89.73%	91.16
R-A1	81.53%	82.05%	81.79
R-A2	61.54%	50.00%	55.17
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	50.00%	66.67
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	57.14%	68.57
R-AM-MNR	28.57%	33.33%	30.77
R-AM-TMP	61.54%	76.92%	68.38
V	97.32%	97.32%	97.32

Table 1: Overall results (top) and detailed results on the WSJ test (bottom) on the closed track of the CoNLL shared task.

denote the best joint labeling of tree t_i, with score $score_{SRL}(L_i|t_i)$ according to our final joint model. Then we choose the labeling L which maximizes:

$$\arg\max_{i\in\{1,\ldots,k\}} \beta \log P(t_i|S) + score_{SRL}(L_i|t_i) \quad (1)$$

Considering top $k = 5$ parse trees using this algorithm resulted in up to 0.4 absolute increase in F-Measure. In future work, we plan to experiment with better ways to combine information from multiple parse trees.

6 Experiments and Results

For our final results we used a joint model with $\alpha = 1.5$ (local model weight), $\beta = 1$ (parse tree log-probability weight) , $N = 15$ (candidate labelings from the local model to consider) , and $k = 5$ (number of alternative parses). The whole training set for the CoNLL-2005 task was used to train the models. It takes about 2 hours to train a local identification model, 40 minutes to train a local classification model, and 7 hours to train a joint re-ranking model.[2]

In Table 1, we present our final development and test results using this model. The percentage of perfectly labeled propositions for the three sets is 55.11% (development), 56.52% (test), and 37.06% (Brown test). The improvement achieved by the joint model relative to the local model is about 2 points absolute in F-Measure, similar to the improvement when gold-standard syntactic parses are used (Toutanova et al., 2005). The relative error reduction is much lower for automatic parses, possibly due to a lower upper bound on performance. It is clear from the drop in performance from the WSJ to Brown test set that our learned model's features do not generalize very well to related domains.

References

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML-2000*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of HLT/NAACL-2004*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-2005*.

[2] On a 3.6GHz machine with 4GB of RAM.

Sparse Bayesian Classification of Predicate Arguments

Richard Johansson and **Pierre Nugues**
LUCAS, Department of Computer Science, Lund University
Box 118
SE-221 00 Lund, Sweden
{richard, pierre}@cs.lth.se

Abstract

We present an application of Sparse Bayesian Learning to the task of semantic role labeling, and we demonstrate that this method produces smaller classifiers than the popular Support Vector approach.

We describe the classification strategy and the features used by the classifier. In particular, the contribution of six parse tree path features is investigated.

1 Introduction

Generalized linear classifiers, in particular Support Vector Machines (SVMs), have recently been successfully applied to the task of semantic role identification and classification (Pradhan et al., 2005), *inter alia*.

Although the SVM approach has a number of properties that make it attractive (above all, excellent software packages exist), it also has drawbacks. First, the resulting classifier is slow since it makes heavy use of kernel function evaluations. This is especially the case in the presence of noise (since each misclassified example has to be stored as a bound support vector). The number of support vectors typically grows with the number of training examples. Although there exist optimization methods that speed up the computations, the main drawback of the SVM approach is still the classification speed.

Another point is that it is necessary to tune the parameters (typically C and γ). This makes it necessary to train repeatedly using cross-validation to find the best combination of parameter values.

Also, the output of the decision function of the SVM is not probabilistic. There are methods to map the decision function onto a probability output using the sigmoid function, but they are considered somewhat ad-hoc (see (Tipping, 2001) for a discussion).

In this paper, we apply a recent learning paradigm, namely *Sparse Bayesian learning*, or more specifically the *Relevance Vector* learning method, to the problem of role classification. Its principal advantages compared to the SVM approach are:

- It typically utilizes fewer examples compared to the SVM, which makes the classifier faster.
- It uses no C parameter, which reduces the need for cross-validation.
- The decision function is adapted for probabilistic output.
- Arbitrary basis functions can be used.

Its significant drawback is that the training procedure relies heavily on dense linear algebra, and is thus difficult to scale up to large training sets and may be prone to numerical difficulties.

For a description of the task and the data, see (Carreras and Màrquez, 2005).

2 Sparse Bayesian Learning and the Relevance Vector Machine

The Sparse Bayesian method is described in detail in (Tipping, 2001). Like other generalized linear learning methods, the resulting binary classifier has the form

$$\text{sign} f(x) = \text{sign} \sum_{i=1}^{m} \alpha_i f_i(x) + b$$

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 177–180, Ann Arbor, June 2005.

where the f_i are basis functions. Training the model then consists of finding a suitable $\boldsymbol{\alpha} = (b, \alpha_1, \ldots, \alpha_m)$ given a data set $(\boldsymbol{X}, \boldsymbol{Y})$.

Analogous with the SVM approach, we can let $f_i(x) = k(x, x_i)$, where x_i is an example from the training set and k a function. We have then arrived at the *Relevance Vector Machine* (RVM). There are however no restrictions on the function k (such as Mercer's condition for SVM). We use the Gaussian kernel $k(x, y) = \exp(-\gamma\|x - y\|^2)$ throughout this work.

We first model the probability of a positive example as a sigmoid applied to $f(x)$. This can be used to write the likelihood function $P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha})$. Instead of a conventional ML approach (maximizing the likelihood with respect to $\boldsymbol{\alpha}$, which would give an overfit model), we now adopt a Bayesian approach and encode the model preferences using priors on $\boldsymbol{\alpha}$. For each α_i, we introduce a parameter s_i and assume that $\alpha_i \in N(0, s_i^{-1})$ (i.e. Gaussian). This is in effect an "Occam penalty" that encodes our preference for sparse models. We should finally specify the distributions of the s_i. However, we make the simplifying assumption that their distribution is flat (noninformative).

We now find the maximum of the *marginal likelihood*, or "evidence", with respect to $\boldsymbol{s}$, that is

$$p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{s}) = \int P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\boldsymbol{s}) d\boldsymbol{\alpha}.$$

This integral is not tractable, hence we approximate the integrand using a Gaussian centered at the mode of the integrand (Laplace's approximation). The marginal likelihood can then be differentiated with respect to $\boldsymbol{s}$, and maximized using iterative methods such as gradient descent.

The algorithm thus proceeds iteratively as follows: First maximize the penalized likelihood function $P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha}) p(\boldsymbol{\alpha}|\boldsymbol{s})$ with respect to $\boldsymbol{\alpha}$ (for example via the Newton-Raphson method), then update the parameters s_i. This goes on until a convergence criterion is met, for example that the s_i changes are small enough. During iteration, the s_i parameters for redundant examples tend to infinity. They (and the corresponding columns of the kernel matrix) are then removed from the model. This is necessary because of numerical stability and also reduces the training time considerably.

We implemented the RVM training method using the ATLAS (Whaley et al., 2000) implementation of the BLAS and LAPACK standard linear algebra APIs. To make the algorithm scale up, we used a working-set strategy that used the results of partial solutions to train the final classifier. Our implementation is based on the original description of the algorithm (Tipping, 2001) rather than the greedy optimized version (Tipping and Faul, 2003), since preliminary experiments suggested a decrease in classification accuracy. Our current implementation can handle training sets up to about 30000 examples.

We used the conventional one-versus-one method for multiclass classification. Although the Sparse Bayesian paradigm is theoretically not limited to binary classifiers, this is of little use in practice, since the size of the Hessian matrix (used while maximizing the likelihood and updating $\boldsymbol{s}$) grows with the number of classes.

3 System Description

Like previous systems for semantic role identification and classification, we used an approach based on classification of nodes in the constituent tree. To simplify training, we used the soft-prune approach as described in (Pradhan et al., 2005), which means that before classification, the nodes were filtered through a binary classifier that classifies them as having a semantic role or not (NON-NULL or NULL). The NULL nodes missed by the filter were included in the training set for the final classifier.

Since our current implementation of the RVM training algorithm does not scale up to large training sets, training on the whole PropBank was infeasible. We instead trained the multiclass classifier on sections 15 – 18, and used an SVM for the soft-pruning classifier, which was then trained on the remaining sections. The excellent LIBSVM (Chang and Lin, 2001) package was used to train the SVM.

The features used by the classifiers can be grouped into predicate and node features. Of the node features, we here pay most attention to the parse tree path features.

3.1 Predicate Features

We used the following predicate features, all of which first appeared in (Gildea and Jurafsky, 2002).

- *Predicate lemma*.
- *Subcategorization frame*.
- *Voice*.

3.2 Node Features

- *Head word* and *head POS*. Like most previous work, we used the head rules of Collins to extract this feature.
- *Position*. A binary feature that describes if the node is before or after the predicate token.
- *Phrase type* (PT), that is the label of the constituent.
- *Named entity*. Type of the first contained NE.
- *Governing category*. As in (Gildea and Jurafsky, 2002), this was used to distinguish subjects from objects. For an NP, this is either S or VP.
- *Path features*. (See next subsection.)

For prepositional phrases, we attached the preposition to the PT and replaced head word and head POS with those of the first contained NP.

3.3 Parse Tree Path Features

Previous studies have shown that the parse tree path feature, used by almost all systems since (Gildea and Jurafsky, 2002), is salient for argument identification. However, it is extremely sparse (which makes the system learn slowly) and is dependent on the quality of the parse tree. We therefore investigated the contribution of the following features in order to come up with a combination of path features that leads to a robust system that generalizes well.

- *Constituent tree path*. As in (Gildea and Jurafsky, 2002), this feature represents the path (consisting of step directions and PTs of the nodes traversed) from the node to the predicate, for example NP↑VP↓VB for a typical object. Removing the direction (as in (Pradhan et al., 2005)) improved neither precision nor recall.
- *Partial path*. To reduce sparsity, we introduced a partial path feature (as in (Pradhan et al., 2005)), which consists of the path from the node to the lowest common ancestor.
- *Dependency tree path*. We believe that labeled dependency paths provide more information about grammatical functions (and, implicitly, semantic relationships) than the raw constituent structure. Since the grammatical functions are not directly available from the parse trees, we investigated two approximations of dependency arc labels: first, the POSs of the head tokens; secondly, the PTs of the head node and its immediate parent (such labels were used in (Ahn et al., 2004)).
- *Shallow path*. Since the UPC shallow parsers were expected to be more robust than the full parsers, we used a shallow path feature. We first built a parse tree using clause and chunk bracketing, and the shallow path feature was then constructed like the constituent tree path.
- *Subpaths*. All subpaths of the constituent path.

We used the parse trees from Charniak's parser to derive all paths except for the shallow path.

4 Results

4.1 Comparison with SVM

The binary classifiers that comprise the one-versus-one multiclass classifier were 89% – 98% smaller when using RVM compared to SVM. However, the performance dropped by about 2 percent. The reason for the drop is possibly that the classifier uses a number of features with extremely sparse distributions (two word features and three path features).

4.2 Path Feature Contributions

To estimate the contribution of each path feature, we measured the difference in performance between a system that used all six features and one where one of the features had been removed. Table 2 shows the results for each of the six features. For the final system, we used the dependency tree path with PT pairs, the shallow path, and the partial path.

4.3 Final System Results

The results of the complete system on the test sets are shown in Table 1. The smaller training set (as mentioned above, we used only sections 15 – 18

	Precision	Recall	$F_{\beta=1}$
Development	73.40%	70.85%	72.10
Test WSJ	75.46%	73.18%	74.30
Test Brown	65.17%	60.59%	62.79
Test WSJ+Brown	74.13%	71.50%	72.79

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	75.46%	73.18%	74.30
A0	84.56%	85.18%	84.87
A1	73.40%	73.35%	73.37
A2	61.99%	57.30%	59.55
A3	71.43%	46.24%	56.14
A4	72.53%	64.71%	68.39
A5	100.00%	40.00%	57.14
AM-ADV	58.13%	51.58%	54.66
AM-CAU	70.59%	49.32%	58.06
AM-DIR	59.62%	36.47%	45.26
AM-DIS	81.79%	71.56%	76.33
AM-EXT	72.22%	40.62%	52.00
AM-LOC	54.05%	55.10%	54.57
AM-MNR	54.33%	52.91%	53.61
AM-MOD	98.52%	96.73%	97.62
AM-NEG	96.96%	96.96%	96.96
AM-PNC	36.75%	37.39%	37.07
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	76.00%	70.19%	72.98
R-A0	83.33%	84.82%	84.07
R-A1	68.75%	70.51%	69.62
R-A2	57.14%	25.00%	34.78
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	92.31%	57.14%	70.59
R-AM-MNR	40.00%	33.33%	36.36
R-AM-TMP	75.00%	69.23%	72.00
V	98.82%	98.82%	98.82

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

for the role classifier) causes the result to be significantly lower than state of the art (F-measure of 79.4, reported in (Pradhan et al., 2005)).

5 Conclusion and Future Work

We have provided an application of Relevance Vector Machines to a large-scale NLP task. The resulting classifiers are drastically smaller that those produced by the SV training methods. On the other hand, the classification accuracy is lower, probably because of the use of lexicalized features.

The results on the Brown test set shows that the genre has a significant impact on the performance.

An evaluation of the contribution of six parse tree path features suggests that dependency tree paths are more useful for semantic role labeling than the traditional constituent tree path.

	P	R	$F_{\beta=1}$
Const. tree	-0.2%	-0.6%	-0.4
Partial	-0.4%	+0.4%	0
Dep. w/ POSs	-0.1%	-0.4%	-0.3
Dep. w/ PT pairs	+0.4%	+0.4%	+0.4
Shallow	-0.1%	+0.4%	+0.1
Const. subpaths	-10.9%	+2.5%	-4.5

Table 2: Contribution of path features

In the future, we will investigate if it is possible to incorporate the γ parameter into the probability model, thus eliminating the need for cross-validation completely. In addition, the training algorithm will need to be redesigned to scale up to larger training sets. The learning paradigm is still young and optimized methods (such as for SVM) have yet to appear. One possible direction is the greedy method described in (Tipping and Faul, 2003).

References

David Ahn, Sisay Fissaha, Valentin Jijkoun, and Maarten de Rijke. 2004. The university of Amsterdam at Senseval-3: Semantic roles and logic forms. In *Proceedings of SENSEVAL-3*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Chih-Chung Chang and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines.*

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James Martin, and Dan Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning*. To appear.

Michael E. Tipping and Anita Faul. 2003. Fast marginal likelihood maximisation for sparse bayesian models. In *9th International Workshop on AI and Statistics*.

Michael E. Tipping. 2001. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211 – 244.

R. Clint Whaley, Antoine Petitet, and Jack J. Dongarra. 2000. Automated empirical optimizations of software and the ATLAS project.

Generalized Inference with Multiple Semantic Role Labeling Systems

Peter Koomen **Vasin Punyakanok** **Dan Roth** **Wen-tau Yih**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{pkoomen2,punyakan,danr,yih}@uiuc.edu

Abstract

We present an approach to semantic role labeling (SRL) that takes the output of multiple argument classifiers and combines them into a coherent predicate-argument output by solving an optimization problem. The optimization stage, which is solved via integer linear programming, takes into account both the recommendation of the classifiers and a set of problem specific constraints, and is thus used both to clean the classification results and to ensure structural integrity of the final role labeling. We illustrate a significant improvement in overall SRL performance through this inference.

1 SRL System Architecture

Our SRL system consists of four stages: *pruning*, *argument identification*, *argument classification*, and *inference*. In particular, the goal of pruning and argument identification is to identify argument candidates for a given verb predicate. The system only classifies the argument candidates into their types during the argument classification stage. Linguistic and structural constraints are incorporated in the inference stage to resolve inconsistent global predictions. The inference stage can take as its input the output of the argument classification of a single system or of multiple systems. We explain the inference for multiple systems in Sec. 2.

1.1 Pruning

Only the constituents in the parse tree are considered as argument candidates. In addition, our system exploits the heuristic introduced by (Xue and Palmer, 2004) to filter out very unlikely constituents. The heuristic is a recursive process starting from the verb whose arguments are to be identified. It first returns the siblings of the verb; then it moves to the parent of the verb, and collects the siblings again. The process goes on until it reaches the root. In addition, if a constituent is a PP (propositional phrase), its children are also collected. Candidates consisting of only a single punctuation mark are not considered.

This heuristic works well with the correct parse trees. However, one of the errors by automatic parsers is due to incorrect PP attachment leading to missing arguments. To attempt to fix this, we consider as arguments the combination of any consecutive NP and PP, and the split of NP and PP inside the NP that was chosen by the previous heuristics.

1.2 Argument Identification

The argument identification stage utilizes binary classification to identify whether a candidate is an argument or not. We train and apply the binary classifiers on the constituents supplied by the pruning stage. Most of the features used in our system are standard features, which include

- **Predicate and POS tag of predicate** indicate the lemma of the predicate and its POS tag.
- **Voice** indicates tbe voice of the predicate.
- **Phrase type** of the constituent.
- **Head word and POS tag of the head word** include head word and its POS tag of the constituent. We use rules introduced by (Collins, 1999) to extract this feature.
- **First and last words and POS tags** of the constituent.
- **Two POS tags before and after** the constituent.
- **Position** feature describes if the constituent is before or after the predicate relative to the position in the sentence.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 181–184, Ann Arbor, June 2005.

- **Path** records the traversal path in the parse tree from the predicate to the constituent.
- **Subcategorization** feature describes the phrase structure around the predicate's parent. It records the immediate structure in the parse tree that expands to its parent.
- **Verb class** feature is the class of the active predicate described in PropBank Frames.
- **Lengths** of the target constituent, in the numbers of words and chunks separately.
- **Chunk** tells if the target argument is, embeds, overlaps, or is embedded in a chunk with its type.
- **Chunk pattern length** feature counts the number of chunks from the predicate to the argument.
- **Clause relative position** is the position of the target word relative to the predicate in the pseudo-parse tree constructed only from clause constituent. There are four configurations—target constituent and predicate share the same parent, target constituent parent is an ancestor of predicate, predicate parent is an ancestor of target word, or otherwise.
- **Clause coverage** describes how much of the local clause (from the predicate) is covered by the argument. It is round to the multiples of $1/4$.

1.3 Argument Classification

This stage assigns the final argument labels to the argument candidates supplied from the previous stage. A multi-class classifier is trained to classify the types of the arguments supplied by the argument identification stage. To reduce the excessive candidates mistakenly output by the previous stage, the classifier can also classify the argument as *NULL* ("not an argument") to discard the argument.

The features used here are the same as those used in the argument identification stage with the following additional features.

- **Syntactic frame** describes the sequential pattern of the noun phrases and the predicate in the sentence. This is the feature introduced by (Xue and Palmer, 2004).
- **Propositional phrase head** is the head of the first phrase after the preposition inside PP.
- **NEG and MOD** feature indicate if the argument is a baseline for AM-NEG or AM-MOD. The rules of the **NEG** and **MOD** features are used in a baseline SRL system developed by Erik Tjong Kim Sang (Carreras and Màrquez, 2004).
- **NE** indicates if the target argument is, embeds, overlaps, or is embedded in a named-entity along with its type.

1.4 Inference

The purpose of this stage is to incorporate some prior linguistic and structural knowledge, such as "arguments do not overlap" or "each verb takes at most one argument of each type." This knowledge is used to resolve any inconsistencies of argument classification in order to generate final legitimate predictions. We use the inference process introduced by (Punyakanok et al., 2004). The process is formulated as an integer linear programming (ILP) problem that takes as inputs the confidences over each type of the arguments supplied by the argument classifier. The output is the optimal solution that maximizes the linear sum of the confidence scores (e.g., the conditional probabilities estimated by the argument classifier), subject to the constraints that encode the domain knowledge.

Formally speaking, the argument classifier attempts to assign labels to a set of arguments, $S^{1:M}$, indexed from 1 to M. Each argument S^i can take any label from a set of argument labels, $\mathcal{P}$, and the indexed set of arguments can take a set of labels, $c^{1:M} \in \mathcal{P}^M$. If we assume that the argument classifier returns an estimated conditional probability distribution, $Prob(S^i = c^i)$, then, given a sentence, the inference procedure seeks an global assignment that maximizes the following objective function,

$$\hat{c}^{1:M} = \underset{c^{1:M} \in \mathcal{P}^M}{\operatorname{argmax}} \sum_{i=1}^{M} Prob(S^i = c^i),$$

subject to linguistic and structural constraints. In other words, this objective function reflects the expected number of correct argument predictions, subject to the constraints. The constraints are encoded as the followings.

- No overlapping or embedding arguments.
- No duplicate argument classes for A0-A5.
- Exactly one V argument per predicate considered.
- If there is C-V, then there has to be a V-A1-CV pattern.
- If there is an R-*arg* argument, then there has to be an *arg* argument.
- If there is a C-*arg* argument, there must be an *arg* argument; moreover, the C-*arg* argument must occur after *arg*.
- Given the predicate, some argument types are illegal (e.g. predicate 'stalk' can take only A0 or A1). The illegal types may consist of A0-A5 and their corresponding C-*arg* and R-*arg* arguments. For each predicate, we look for the minimum value of i such that the class Ai is mentioned in its frame file as well as its maximum value j. All argument types Ak such that $k < i$ or $k > j$ are considered illegal.

2 Inference with Multiple SRL Systems

The inference process allows a natural way to combine the outputs from multiple argument classifiers. Specifically, given k argument classifiers which perform classification on k argument sets, $\{S_1, \ldots, S_k\}$. The inference process aims to optimize the objective function:

$$\hat{c}^{1:N} = \underset{c^{1:N} \in \mathcal{P}^N}{\operatorname{argmax}} \sum_{i=1}^{N} Prob(S^i = c^i),$$

where $S^{1:N} = \bigcup_{i=1}^{k} S_i$, and

$$Prob(S^i = c^i) = \frac{1}{k} \sum_{j=1}^{k} Prob_j(S^i = c^i),$$

where $Prob_j$ is the probability output by system j.

Note that all systems may not output with the same set of argument candidates due to the pruning and argument identification. For the systems that do not output for any candidate, we assign the probability with a prior to this *phantom* candidate. In particular, the probability of the *NULL* class is set to be 0.6 based on empirical tests, and the probabilities of the other classes are set proportionally to their occurrence frequencies in the training data.

For example, Figure 1 shows the two candidate sets for a fragment of a sentence, "*..., traders say, unable to* ***cool*** *the selling panic in both stocks and futures.*" In this example, system A has two argument candidates, a_1 = "traders" and a_4 = "the selling panic in both stocks and futures"; system B has three argument candidates, b_1 = "traders", b_2 = "the selling panic", and b_3 = "in both stocks and futures". The phantom candidates are created for a_2, a_3, and b_4 of which probability is set to the prior.

Specifically for this implementation, we first train two SRL systems that use Collins' parser and Charniak's parser respectively. In fact, these two parsers have noticeably different output. In evaluation, we run the system that was trained with Charniak's parser 5 times with the top-5 parse trees output by Charniak's parser[1]. Together we have six different outputs per predicate. Per each parse tree output, we ran the first three stages, namely pruning, argument identification, and argument classification. Then a joint inference stage is used to resolve the inconsistency of the output of argument classification in these systems.

[1]The top parse tree were from the official output by CoNLL. The 2nd-5th parse trees were output by Charniak's parser.

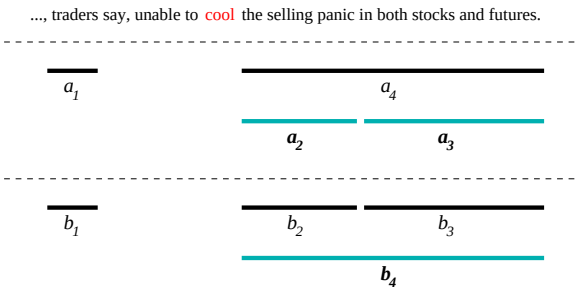


Figure 1: Two SRL systems' output (a_1, a_4, b_1, b_2, and b_3), and phantom candidates (a_2, a_3, and b_4).

3 Learning and Evaluation

The learning algorithm used is a variation of the Winnow update rule incorporated in SNoW (Roth, 1998; Roth and Yih, 2002), a multi-class classifier that is tailored for large scale learning tasks. SNoW learns a sparse network of linear functions, in which the targets (argument border predictions or argument type predictions, in this case) are represented as linear functions over a common feature space. It improves the basic Winnow multiplicative update rule with a regularization term, which has the effect of trying to separate the data with a large margin separator (Grove and Roth, 2001; Hang et al., 2002) and voted (averaged) weight vector (Freund and Schapire, 1999).

Softmax function (Bishop, 1995) is used to convert raw activation to conditional probabilities. If there are n classes and the raw activation of class i is act_i, the posterior estimation for class i is

$$Prob(i) = \frac{e^{act_i}}{\sum_{1 \le j \le n} e^{act_j}}.$$

In summary, training used both full and partial syntactic information as described in Section 1. In training, SNoW's default parameters were used with the exception of the separator thickness 1.5, the use of average weight vector, and 5 training cycles. The parameters are optimized on the development set.

Training for each system took about 6 hours. The evaluation on both test sets which included running

	Precision	Recall	$F_{\beta=1}$
Development	80.05%	74.83%	77.35
Test WSJ	82.28%	76.78%	79.44
Test Brown	73.38%	62.93%	67.75
Test WSJ+Brown	81.18%	74.92%	77.92

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	82.28%	76.78%	79.44
A0	88.22%	87.88%	88.05
A1	82.25%	77.69%	79.91
A2	78.27%	60.36%	68.16
A3	82.73%	52.60%	64.31
A4	83.91%	71.57%	77.25
A5	0.00%	0.00%	0.00
AM-ADV	63.82%	56.13%	59.73
AM-CAU	64.15%	46.58%	53.97
AM-DIR	57.89%	38.82%	46.48
AM-DIS	75.44%	80.62%	77.95
AM-EXT	68.18%	46.88%	55.56
AM-LOC	66.67%	55.10%	60.33
AM-MNR	66.79%	53.20%	59.22
AM-MOD	96.11%	98.73%	97.40
AM-NEG	97.40%	97.83%	97.61
AM-PNC	60.00%	36.52%	45.41
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	78.16%	76.72%	77.44
R-A0	89.72%	85.71%	87.67
R-A1	70.00%	76.28%	73.01
R-A2	85.71%	37.50%	52.17
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	57.14%	68.57
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	72.34%	65.38%	68.69
V	98.92%	97.10%	98.00

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

with all six different parse trees (assumed already given) and the joint inference took about 4.5 hours.

	Precision	Recall	$F_{\beta=1}$
Charniak-1	75.40%	74.13%	74.76
Charniak-2	74.21%	73.06%	73.63
Charniak-3	73.52%	72.31%	72.91
Charniak-4	74.29%	72.92%	73.60
Charniak-5	72.57%	71.40%	71.98
Collins	73.89%	70.11%	71.95
Joint inference	80.05%	74.83%	77.35

Table 2: The results of individual systems and the result with joint inference on the development set.

Overall results on the development and test sets are shown in Table 1. Table 2 shows the results of individual systems and the improvement gained by the joint inference on the development set.

4 Conclusions

We present an implementation of SRL system which composed of four stages—1) pruning, 2) argument identification, 3) argument classification, and 4) inference. The inference provides a natural way to take the output of multiple argument classifiers and combines them into a coherent predicate-argument output. Significant improvement in overall SRL performance through this inference is illustrated.

Acknowledgments

We are grateful to Dash Optimization for the free academic use of Xpress-MP. This research is supported by ARDA's AQUAINT Program, DOI's Reflex program, and an ONR MURI Award.

References

C. Bishop, 1995. *Neural Networks for Pattern Recognition*, chapter 6.4: Modelling conditional distributions, page 215. Oxford University Press.

X. Carreras and L. Màrquez. 2004. Introduction to the conll-2004 shared tasks: Semantic role labeling. In *Proc. of CoNLL-2004*.

M. Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, Computer Science Department, University of Pennsylvenia, Philadelphia.

Y. Freund and R. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

A. Grove and D. Roth. 2001. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.

T. Hang, F. Damerau, and D. Johnson. 2002. Text chunking based on a generalization of winnow. *Journal of Machine Learning Research*, 2:615–637.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proc. of COLING-2004*.

D. Roth and W. Yih. 2002. Probabilistic reasoning for entity & relation recognition. In *Proc. of COLING-2002*, pages 835–841.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of AAAI*, pages 806–813.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. of the EMNLP-2004*, pages 88–94, Barcelona, Spain.

Semantic Role Labeling via Consensus in Pattern-Matching

Chi-San (Althon) Lin
Department of Computer Science
Waikato University
Hamilton, New Zealand
cl123@cs.waikato.ac.nz

Tony C. Smith
Department of Computer Science
Waikato University
Hamilton, New Zealand
tcs@cs.waikato.ac.nz

Abstract

This paper describes a system for semantic role labeling for the CoNLL2005 Shared task. We divide the task into two sub-tasks: boundary recognition by a general tree-based predicate-argument recognition algorithm to convert a parse tree into a flat representation of all predicates and their related boundaries, and role labeling by a consensus model using a pattern-matching framework to find suitable roles for core constituents and adjuncts. We describe the system architecture and report results for the CoNLL2005 development dataset.

1 Introduction

Semantic role labeling is to find all arguments for all predicates in a sentence, and classify them by semantic roles such as A0, A1, AM-TMP and so on. The performance of semantic role labeling can play a key role in Natural Language Processing applications, such as Information Extraction, Question Answering, and Summarization (Pradhan et al., 2004).

Most existing systems separate semantic role labeling into two sub-problems, boundary recognition and role classification, and use feature-based models to address both (Carreras et al., 2004). Our strategy is to develop a boundary analyzer by a general tree-based predicate-argument recognition algorithm (GT-PARA) for boundary recognition, and a pattern-matching model for role classification. The only information used in our system is Charniak's annotation with words, which contains all useful syntactic annotations. Five features, which are Headword, Phrase type, Voice, Target verb, and Preposition (of the first word), and a Pattern set, which includes numbers and types of roles in a pattern, are used for the pattern-matching approach. We develop a Pattern Database, trained by Wall Street Journal section 02 to 21, as our knowledge/Data base. The system outline is described in the following section.

2 System Description

An overview of the system architecture is shown in Figure 1. The input is a full parse tree for each sentence. We convert a sentence with words, and Charniak's information into a parsed tree as the input of GT-PARA. GT-PARA then converts the parse tree into a flat representation with all predicates and arguments expressed in [**GPLVR**] format; where

G: Grammatical function – 5 denotes subject, 3 object, and 2 others;
P: Phrase type of this boundary – 00 denotes ADJP, 01 ADVP, 02 NP, 03 PP, 04 S, 05 SBAR, 06 SBARQ, 07 SINV, 08 SQ, 09 VP, 10 WHADVP, 11 WHNP, 12 WHPP, and 13 Others
L: Distance (and position) of the argument with respect to the predicate that follows
V: Voice of the predicate, 0: active 1: passive
R: Distance (and position) of the argument with respect to the preceding predicate (n.b. **L** and **R** are mutually exclusive).

An example of the output of GT-PARA is shown in Figure 2. There is one predicate "take" in the sample input sentence. There are 4 arguments for that predicate, denoted as "302110", "AM-MOD", "203011", and "302012" respectively. "302110" symbolizes the NP Object of distance 1 prior to the passive predicate. "203011" symbolizes an undefined PP argument (which

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 185–188, Ann Arbor, June 2005.

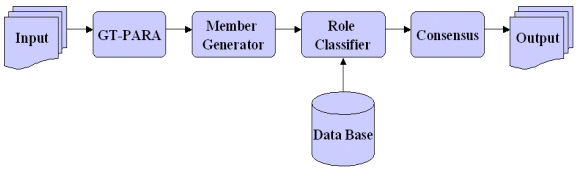


Figure 1: System Architecture

means it can be a core argument or an adjunct) with distance 1 after the passive predicate. And "302012" symbolizes a NP Object with distance 2 after the passive predicate.

For all boundaries extracted by GT-PARA, we simply denote all boundaries with noun phrases (NP) or similar phrases, such as WHNP, SBAR, and so on, as *core pattern candidates* and all boundaries with prepositional phrases (PP), ADJP, ADVP, or similar phrases, such as WHADJP, WHADVP, and so on, as *adjunct candidates*. But there is no exact rule for defining a core role or an adjunct explicitly in a boundary span, for example, given a sentence where

(1) P1 is done by P2. (P1 and P2 are two groups of words or phrases)

We can guess P1 might be labeled with "A1", and P2 with "A0" if there is no further feature information. But if the "head word" feature of P2 is "hour", for example, P2 can be labeled with "AM-TMP" instead. Because there are some uncertainties between core roles and adjuncts before labeling, we use the Member Generator (in Figure 1) to create all possible combinations, called members, from the output of GT-PARA by changing ANs (Core Role Candidates) into AMs (Adjunct Candidates), or AMs into ANs, except core candidates before predicates. All possible combinations (members) for the example in Figure 1 are

M1: [AN1, AM-MOD, V, AM1<points>(from), AN2] (original)
M2: [AN1 AM-MOD V *AN3* (from) AN2]
(change AM1 as AN3)
M3: [AN1 AM-MOD V AM1<point>(from) *AM2*<week>] (change AN2 as AM2)
M4: [AN1 AM-MOD V *AN3*<point>(from) *AM2*<week>]
(change AM1 as AN3 and one AN2 as AM2)

The output from the Member Generator is passed to the Role Classifier, which finds all possible roles for each member with suitable core

Words	POS	Full Tree Syntax	Predicate	Boundaries
The	DT	(S1(S(NP(NP*	-	(302110*
economy	NN	*	-	*
's	POS	*)	-	*
temperature	NN	*)	-	*)
will	MD	(VP*	-	(AM-MOD*)
be	AUX	(VP*	-	*
taken	VBN	(VP*	take	(V*V)
from	IN	(PP*	-	(203011*
several	JJ	(NP*	-	*
vantage	NN	*	-	*
points	NNS	*))	-	*)
this	DT	(NP*	-	(302012*
week	NN	*)	-	*)
.	.	*))	-	*

Figure 2: Illustration of an output of GT-PARA of a sentence, "The economy 's temperature will be taken from several vantage points this week."

roles and adjuncts according to a Database built up by training data, in which each predicate has different patterns associated with it, each pattern has different semantic roles, and each role has the following format.

Role {Phrase type} < Head Word> (preposition)

There is an additional Boolean voice for a predicate to show if the predicate is passive or active (0: denotes active, 1: denotes passive). Each pattern includes a count on the number of the same patterns learned from the training data (denoted as "[*statistical figure*]"). For example, eight patterns for a predicate lemma "take" are

1. [30] A0{NP}<buyers> V{VP}<take>-0 A1{NP}<stake>
2. [1] A0{NP}<U.S.> V{VP}<take>-0 A1{NP}<%> A2{PP}<Canada>(from) AM-ADV{ADVP}<up>(up)
3. [2] A0{NP}<Confidence> V{VP}<take>-0 A1{NP}<dive> AM-ADV{SBAR}< figures>(if)
4. [1] A1{NP}<it> AM-MOD{VP}<could> V{VP}<take>-0 A2{NP}<place> AM-TMP{NP}<today> AM-LOC{PP}<Express>(at)
5. [1] AM-TMP{NP}< week> A0{NP}<government> V{VP}<take>-0 A1{NP}<bills> AM-DIR{PP}<to>(to)
6. [3] A1{NP}<cells> V{VP}<take>-1 A2{PP}<tissue>(from)
7. [6] A1{NP}<action> V{VP}<take>-1
8. [1] AM-TMP{ADVP}<far> A1{NP}<festivities> V{VP}<take>-1 AM-EXT{PP}<entirely> A0{NP}<eating>(by)

Role Classifier consists of two parts, AN classifier and AM classifier, which process core argu-

ments and adjuncts respectively. AN classifier finds a suitable core pattern for labeled core pattern candidates in each member generated by Member Generator according to

(1) the same numbers of core roles
(2) the same **prepositions** for each core role
(3) the same **phrase types** for each core role
(4) the same voice (active or passive)

AM classifier finds a suitable adjunct role for any labeled adjunct candidate in each member generated by Member Generator according to

(1) the same Head Word
(2) the same Phrase type
(3) the highest statistical probability learned from the training data

The followings are the results for each member after Role Classification

M1: [AN1, AM-MOD, V, AM1<points>(from), AN2] (no pattern applied)

M2: [AN1 AM-MOD V AN1 (from) AN2] (no pattern applied)

M3: [*A1* AM-MOD V AM1<point>(from) *AM-TMP*<week>] (ANs by pattern 7, AM-TMP by pattern 5) [stat: 6]

M4: [*A1* AM-MOD V *A2* (from) *AM-TMP*<week>] (ANs by pattern 6, AM-TMP by pattern 5) [stat: 3]

Decision-making in the Consensus component (see Figure 1) handles the final selection by selecting the highest score using the following formula.

$\text{Score}_k = (\alpha_1 * R_k + \alpha_2 * V_k + \alpha_3 * S_k)$ for each X_k (k=1 .. K, generated by Member Generator and Role Classifier), where

R_k : numbers of all roles being labeled
V_k : votes of a pattern with the same roles
S_k : statistical figure learned from trained data
X_k : different pattern by Member General and Role Classifier

α_1, α_2, and α_3 are weights ($\alpha_1 >> \alpha_2 >> \alpha_3$) used to rank the relative contribution of R_k, V_k, and S_k. Empirical studies led to the use of a so-called Max-labeled-role Heuristic to derive suitable values for these weights.

The final consensus decision for role classification is determined by calculating

$$\textbf{Consensus} = \max_{k=1}^{K} \text{Score}_k$$

There are 3 roles labeled in M3, which are AN1 as A1, AM-MOD, AM2 as AM-TMP respectively. And there are 4 roles labeled in M4, which are AN1 as A1, AM-MOD, AN3 as A2, and AM2 as AM-TMP respectively. Consensus scores for M3, and M4 are

$(\alpha_1 * 3 + \alpha_2 * 1 + \alpha_3 * 6)$, and
$(\alpha_1 * 4 + \alpha_2 * 1 + \alpha_3 * 3)$.

So the pattern [**A1** AM-MOD V **A2**(from) **AM-TMP**<week>] in M4 applied by Pattern 6 and Pattern 5 is selected due to the most roles labeled.

3 Data and Evaluation

We extracted patterns from the training data (WSJ Section 02 to 21) to build up a pattern database. Table 1 reveals sparseness of the pattern database. Twenty-six percent of predicates contain only one pattern, and fifteen two patterns. Seventy-five percents of predicates contain no more than 10 patterns.

No	1	2	3	4	5	5-10	11-50	51-100	>100
%	26	15	10	7	5	13	20	4	2
A %	26	40	50	57	62	75	94	98	100

Table 1: Statistical figures on the number of patterns collected from training, WSJ Section 02-21

The evaluation software, *srl-eval.pl*, is available from CoNLL2005 Shared Task[1], which is the official script for evaluation of CoNLL-2005 Shared Task systems. In order to test boundary performance of GT-PARA, we simply convert all correct propositional arguments into A0s, except AM-MOD and AM-NEG for both the training dataset (WSJ Sections 15-18) and the development dataset (WSJ Section 24).

4 Experimental Results

The results of classification on the development, and test data of the CoNLL2005 shared task are outlined in Table 2. The overall results on the Development, Test-WSJ, Test-Brown, and Test-WSJ+Brown datasets for F-score are 65.78, 67.91, 58.58 and 66.72 respectively, which are moderate compared to the best result reported in CoNLL2004 Shared Task (Carreras et al., 2004) using partial trees and the result in (Pradhan et al., 2004). The results for boundary recognition via GT-PARA are summarized in Table 3.

[1] http://www.lsi.upc.edu/~srlconll/soft.html

	Precision	Recall	$F_{\beta=1}$
Development(WSJ24)	70.11%	61.96%	65.78
Test WSJ	71.49%	64.67%	67.91
Test Brown	65.75%	52.82%	58.58
Test WSJ + Brown	70.80%	63.09%	66.72

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	71.49%	64.67%	67.91
A0	81.74%	81.53%	81.64
A1	71.61%	69.54%	70.56
A2	63.73%	40.36%	49.42
A3	68.60%	34.10%	45.56
A4	33.93%	18.63%	24.05
A5	0.00%	0.00%	0.00
AA	0.00%	0.00%	0.00
AM-ADV	36.26%	31.82%	33.89
AM-CAU	52.00%	35.62%	42.28
AM-DIR	20.11%	42.35%	27.27
AM-DIS	73.91%	63.75%	68.46
AM-EXT	12.90%	12.50%	12.70
AM-LOC	60.80%	33.33%	43.06
AM-MNR	43.57%	30.52%	35.90
AM-MOD	99.21%	90.93%	94.89
AM-NEG	96.38%	92.61%	94.46
AM-PNC	13.69%	31.30%	19.05
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	71.62%	54.55%	61.93
R-A0	93.37%	69.20%	79.49
R-A1	82.24%	56.41%	66.92
R-A2	100.00%	25.00%	40.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	0.00%	0.00%	0.00
V	97.34%	95.25%	96.29

Table 2: Overall results (top) and detailed results on the WSJ test (bottom), obtained by the system.

The overall performance (F1: 76.43) on the WSJ Section 24 is not as good as on the WSJ Section 21 (F1: 85.78). The poor performance for the development was caused by more parser errors in the WSJ Section 24. Most parser errors are brought on by continuous phrases with commas and/or quotation marks.

One interesting fact is that when we tested our system using the data in CoNLL2004 shared task, we found the result with the train data WSJ 15-18 on the WSJ 21 is 73.48 shown in Table 4, which increases about 7 points in the F1 score, compared to WSJ 24 shown in Table 2. We found the labeling accuracy for WSJ 24 is 87.73, which is close to 89.30 for WSJ Section 21. But the results of boundary recognition in Table 3 for the two data are 9.14 points different, which leads to the better performance in WSJ Section 21. Boundary recognition as mentioned in CoNLL004 does play a very important role in this system as well.

	Precision	Recall	$F_{\beta=1}$
WSJ 15-18	87.23%	83.98%	85.57
WSJ 21	86.89%	84.70%	85.78
WSJ 24	78.88%	74.12%	76.43

Table 3: Boundary Recognition results by GT-PARA on WSJ 15-18, WSJ 21 and WSJ 24 sets

WSJ 21	Precision	Recall	$F_{\beta=1}$
Overall	78.06%	69.41%	73.48

Table 4: System results by the training data WSJ 15-18 on the WSJ Section 21

5 Conclusion

We have described a semantic role labeling architecture via consensus in a pattern-matching system. The pattern-matching system is based on linear pattern matching utilising statistical consensus for decision-making. A General Tree-based Predicate-Argument Boundary Recognition Algorithm (GT-PARA) handles the conversion process, turning a parse tree into a flat representation with all predicates and their arguments labeled with some useful features, such as phrase types. Label accuracy of Consensus model for role classification is stable but performance results of GT-PARA vary on different datasets, which is the key role for the overall results. Although the results seem moderate on test data, this system offers a decidedly different approach to the problem of semantic role labeling.

References

Xavier Carreras, Lluís Màrquez and Grzegorz Chrupała. 2004. Hierarchical Recognition of Propositional Arguments with Perceptrons. In *Proceeding of CoNLL'2004 Shared Task*.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J., Jurafsky, D. 2004. " Shallow Semantic Parsing using Support Vector Machines ", in *Proceedings of HLT/NAACL-2004*, Boston, MA.

Semantic Role Labeing System using Maximum Entropy Classifier *

Ting Liu, Wanxiang Che, Sheng Li, Yuxuan Hu and Huaijun Liu
Information Retrieval Lab
School of Computer Science and Technology
Harbin Institute of Technology
China, 150001
{tliu, car, ls, yxhu, hjliu}@ir.hit.edu.cn

Abstract

A maximum entropy classifier is used in our semantic role labeling system, which takes syntactic constituents as the labeling units. The maximum entropy classifier is trained to identify and classify the predicates' semantic arguments together. Only the constituents with the largest probability among embedding ones are kept. After predicting all arguments which have matching constituents in full parsing trees, a simple rule-based post-processing is applied to correct the arguments which have no matching constituents in these trees. Some useful features and their combinations are evaluated.

1 Introduction

The semantic role labeling (SRL) is to assign syntactic constituents with semantic roles (arguments) of predicates (most frequently verbs) in sentences. A semantic role is the relationship that a syntactic constituent has with a predicate. Typical semantic arguments include Agent, Patient, Instrument, etc. and also adjunctive arguments indicating Locative, Temporal, Manner, Cause, etc. It can be used in lots of natural language processing application systems in which some kind of semantic interpretation is needed, such as question and answering, information extraction, machine translation, paraphrasing, and so on.

*This research was supported by National Natural Science Foundation of China via grant 60435020

Last year, CoNLL-2004 hold a semantic role labeling shared task (Carreras and Màrquez, 2004) to test the participant systems' performance based on shallow syntactic parser results. In 2005, SRL shared task is continued (Carreras and Màrquez, 2005), because it is a complex task and now it is far from desired performance.

In our SRL system, we select maximum entropy (Berger et al., 1996) as a classifier to implement the semantic role labeling system. Different from the best classifier reported in literatures (Pradhan et al., 2005) – support vector machines (SVMs) (Vapnik, 1995), it is much easier for maximum entropy classifier to handle the multi-class classification problem without additional post-processing steps. The classifier is much faster than training SVMs classifiers. In addition, maximum entropy classifier can be tuned to minimize over-fitting by adjusting gaussian prior. Xue and Palmer (2004; 2005) and Kwon et al. (2004) have applied the maximum entropy classifier to semantic role labeling task successfully.

In the following sections, we will describe our system and report our results on development and test sets.

2 System Description

2.1 Constituent-by-Constituent

We use syntactic constituent as the unit of labeling. However, it is impossible for each argument to find its matching constituent in all auto parsing trees. According to statistics, about 10% arguments have no matching constituents in the training set of 245,353

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 189–192, Ann Arbor, June 2005.

constituents. The top five arguments with no matching constituents are shown in Table 1. Here, Charniak parser got 10.08% no matching arguments and Collins parser got 11.89%.

Table 1: The top five arguments with no matching constituents.

Args	Cha parser	Col parser	Both
AM-MOD	9179	9205	9153
A1	5496	7273	3822
AM-NEG	3200	3217	3185
AM-DIS	1451	1482	1404
A0	1416	2811	925

Therefore, we can see that Charniak parser got a better result than Collins parser in the task of SRL. So we use the full analysis results created by Charniak parser as our classifier's inputs. Assume that we could label all AM-MOD and AM-NEG arguments correctly with simple post processing rules, the upper bound of performance could achieve about 95% recall.

At the same time, we can see that for some arguments, both parsers got lots of no matchings such as AM-MOD, AM-NEG, and so on. After analyzing the training data, we can recognize that the performance of these arguments can improve a lot after using some simple post processing rules only, however other arguments' no matching are caused primarily by parsing errors. The comparison between using and not using post processing rules is shown in Section 3.2.

Because of the high speed and no affection in the number of classes with efficiency of maximum entropy classifier, we just use one stage to label all arguments of predicates. It means that the "NULL" tag of constituents is regarded as a class like "ArgN" and "ArgM".

2.2 Features

The following features, which we refer to as the basic features modified lightly from Pradhan et al. (2005), are provided in the shared task data for each constituent.

- **Predicate lemma**
- **Path:** The syntactic path through the parse tree from the parse constituent to the predicate.
- **Phrase type**
- **Position:** The position of the constituent with respect to its predicate. It has two values, "before" and "after", for the predicate. For the situation of "cover", we use a heuristic rule to ignore all of them because there is no chance for them to become an argument of the predicate.
- **Voice:** Whether the predicate is realized as an active or passive construction. We use a simple rule to recognize passive voiced predicates which are labeled with part of speech – VBN and sequences with AUX.
- **Head word stem:** The stemming result of the constituent's syntactic head. A rule based stemming algorithm (Porter, 1980) is used. Collins Ph.D thesis (Collins, 1999)[Appendix. A] describs some rules to identify the head word of a constituent. Especially for prepositional phrase (PP) constituent, the normal head words are not very discriminative. So we use the last noun in the PP replacing the traditional head word.
- **Sub-categorization**

We also use the following additional features.

- **Predicate POS**
- **Predicate suffix:** The suffix of the predicate. Here, we use the last 3 characters as the feature.
- **Named entity:** The named entity's type in the constituent if it ends with a named entity. There are four types: LOC, ORG, PER and MISC.
- **Path length:** The length of the path between a constituent and its predicate.
- **Partial path:** The part of the path from the constituent to the lowest common ancestor of the predicate and the constituent.
- **Clause layer:** The number of clauses on the path between a constituent and its predicate.
- **Head word POS**
- **Last word stem:** The stemming result of the last word of the constituent.
- **Last word POS**

We also use some combinations of the above features to build some combinational features. Lots of combinational features which were supposed to contribute the SRL task of added one by one. At the same time, we removed ones which made the performance decrease in practical experiments. At last, we keep the following combinations:

- Position + Voice
- Path length + Clause layer
- Predicate + Path
- Path + Position + Voice
- Path + Position + Voice + Predicate
- Head word stem + Predicate
- Head word stem + Predicate + Path
- Head word stem + Phrase
- Clause layer + Position + Predicate

All of the features and their combinations are used without feature filtering strategy.

2.3 Classifier

Le Zhang's Maximum Entropy Modeling Toolkit [1], and the L-BFGS parameter estimation algorithm with gaussian prior smoothing (Chen and Rosenfeld, 1999) are used as the maximum entropy classifier. We set gaussian prior to be 2 and use 1,000 iterations in the toolkit to get an optimal result through some comparative experiments.

2.4 No Embedding

The system described above might label two constituents even if one embeds in another, which is not allowed by the SRL rule. So we keep only one argument when more arguments embedding happens. Because it is easy for maximum entropy classifier to output each prediction's probability, we can label the constituent which has the largest probability among the embedding ones.

2.5 Post Processing Stage

After labeling the arguments which are matched with constituents exactly, we have to handle the arguments, such as `AM-MOD`, `AM-NEG` and `AM-DIS`, which have few matching with the constituents described in Section 2.1. So a post processing is given by using some simply rules:

- Tag target verb and successive particles as V.
- Tag "not" and "n't" in target verb chunk as `AM-NEG`.
- Tag modal verbs in target verb chunk, such as words with POS of "MD", "going to", and so on, as `AM-MOD`.
- Tag the words with POS of "CC" and "RB" at the start of a clause which include the target verb as `AM-DIS`.

3 Experiments

3.1 Data and Evaluation Metrics

The data provided for the shared task is a part of PropBank corpus. It consists of the sections from the Wall Street Journal part of Penn Treebank. Sections 02-21 are training sets, and Section 24 is development set. The results are evaluated for precision, recall and $F_{\beta=1}$ numbers using the *srl-eval.pl* script provided by the shared task organizers.

3.2 Post Processing

After using post processing rules, the final $F_{\beta=1}$ is improved from 71.02% to 75.27%.

[1] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

3.3 Performance Curve

Because the training corpus is substantially enlarged, this allows us to test the scalability of learning-based SRL systems to large data set and compute learning curves to see how many data are necessary to train. We divide the training set, 20 sections Penn Treebank into 5 parts with 4 sections in each part. There are about 8,000 sentences in each part. Figure 1 shows the change of performance as a function of training set size. When all of training data are used, we get the best system performance as described in Section 3.4.

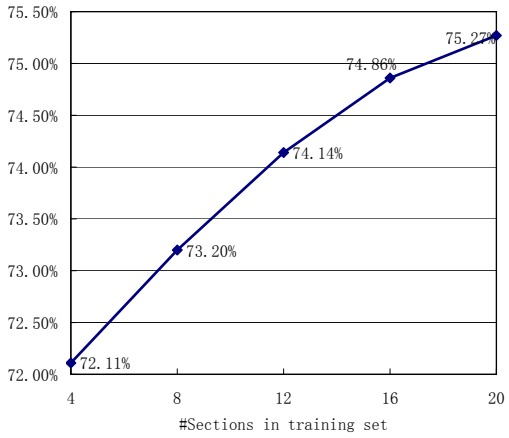


Figure 1: Our SRL system performance curve (of $F_{\beta=1}$) effecting of the training set size.

We can see that as the training set becomes larger and larger, so does the performance of SRL system. However, the rate of increase slackens. So we can say that at present state, the larger training data has favorable effect on the improvement of SRL system performance.

3.4 Best System Results

In all the experiments, all of the features and their combinations described above are used in our system. Table 2 presents our best system performance on the development and test sets.

From the test results, we can see that our system gets much worse performance on Brown corpus than WSJ corpus. The reason is easy to be understood for the dropping of automatic syntactic parser performance on new corpus but WSJ corpus.

The training time on PIV 2.4G CPU and 1G Mem machine is about 20 hours on all 20 sections, 39,832-

	Precision	Recall	$F_{\beta=1}$
Development	79.65%	71.34%	75.27
Test WSJ	80.48%	72.79%	76.44
Test Brown	71.13%	59.99%	65.09
Test WSJ+Brown	79.30%	71.08%	74.97

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	80.48%	72.79%	76.44
A0	88.14%	83.61%	85.81
A1	79.62%	72.88%	76.10
A2	73.67%	65.05%	69.09
A3	76.03%	53.18%	62.59
A4	78.02%	69.61%	73.58
A5	100.00%	40.00%	57.14
AM-ADV	59.85%	48.02%	53.29
AM-CAU	68.18%	41.10%	51.28
AM-DIR	56.60%	35.29%	43.48
AM-DIS	76.32%	72.50%	74.36
AM-EXT	83.33%	46.88%	60.00
AM-LOC	65.31%	52.89%	58.45
AM-MNR	58.28%	51.16%	54.49
AM-MOD	98.52%	96.37%	97.43
AM-NEG	97.79%	96.09%	96.93
AM-PNC	43.68%	33.04%	37.62
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	78.38%	66.70%	72.07
R-A0	81.70%	85.71%	83.66
R-A1	77.62%	71.15%	74.25
R-A2	60.00%	37.50%	46.15
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	83.33%	47.62%	60.61
R-AM-MNR	66.67%	33.33%	44.44
R-AM-TMP	77.27%	65.38%	70.83
V	98.71%	98.71%	98.71

Table 2: Overall results (top) and detailed results on the WSJ test (bottom).

sentences training set with 1,000 iterations and more than 1.5 million samples and 2 million features. The predicting time is about 160 seconds on 1,346-sentences development set.

4 Conclusions

We have described a maximum entropy classifier is our semantic role labeling system, which takes syntactic constituents as the labeling units. The fast training speed of the maximum entropy classifier allows us just use one stage of arguments identification and classification to build the system. Some useful features and their combinations are evaluated. Only the constituents with the largest probability among embedding ones are kept. After predicting all arguments which have matching constituents in full parsing trees, a simple rule-based post-processing is applied to correct the arguments which have no matching constituents. The constituent-based method depends much on the syntactic parsing performance. The comparison between WSJ and Brown test sets results fully demonstrates the point of view.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97, Boston, MA, USA.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Pennsylvania University.

Namhee Kwon, Michael Fleischman, and Eduard Hovy. 2004. Framenet-based semantic parsing using maximum entropy models. In *Proc. Coling 2004*.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *Machine Learning Journal*.

Vladamir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proc. EMNLP 2004*.

Nianwen Xue and Martha Palmer. 2005. Automatic semantic role labeling for chinese verbs. In *Proc. IJCAI 2005*.

Semantic Role Labeling as Sequential Tagging

Lluís Màrquez, **Pere Comas**, **Jesús Giménez** and **Neus Català**
TALP Research Centre
Technical University of Catalonia (UPC)
{lluism,pcomas,jgimenez,ncatala}@lsi.upc.edu

Abstract

In this paper we present a semantic role labeling system submitted to the CoNLL-2005 shared task. The system makes use of partial and full syntactic information and converts the task into a sequential BIO-tagging. As a result, the labeling architecture is very simple . Building on a state-of-the-art set of features, a binary classifier for each label is trained using AdaBoost with fixed depth decision trees. The final system, which combines the outputs of two base systems performed F_1=76.59 on the official test set. Additionally, we provide results comparing the system when using partial vs. full parsing input information.

1 Goals and System Architecture

The goal of our work is twofold. On the one hand, we want to test whether it is possible to implement a competitive SRL system by reducing the task to a sequential tagging. On the other hand, we want to investigate the effect of replacing partial parsing information by full parsing. For that, we built two different individual systems with a shared sequential strategy but using UPC chunks-clauses, and Charniak's parses, respectively. We will refer to those systems as $\mathrm{PP_{UPC}}$ and $\mathrm{FP_{CHA}}$, hereinafter.

Both partial and full parsing annotations provided as input information are of hierarchical nature. Our system navigates through these syntactic structures in order to select a subset of constituents organized sequentially (i.e., non embedding). Propositions are treated independently, that is, each target verb generates a sequence of tokens to be annotated. We call this pre-processing step **sequentialization**.

The sequential tokens are selected by exploring the sentence spans or regions defined by the clause boundaries[1]. The top-most syntactic constituents falling inside these regions are selected as tokens. Note that this strategy is independent of the input syntactic annotation explored, provided it contains clause boundaries. It happens that, in the case of full parses, this node selection strategy is equivalent to the pruning process defined by Xue and Palmer (2004), which selects sibling nodes along the path of ancestors from the verb predicate to the root of the tree[2]. Due to this pruning stage, the upper-bound recall figures are 95.67% for $\mathrm{PP_{UPC}}$ and 90.32% for $\mathrm{FP_{CHA}}$. These values give F_1 performance upper bounds of 97.79 and 94.91, respectively, assuming perfect predictors (100% precision).

The nodes selected are labeled with B-I-O tags depending if they are at the beginning, inside, or outside of a verb argument. There is a total of 37 argument types, which amount to 37*2+1=75 labels.

Regarding the **learning algorithm**, we used generalized AdaBoost with real-valued weak classifiers, which constructs an ensemble of decision trees of fixed depth (Schapire and Singer, 1999). We considered a one-vs-all decomposition into binary prob-

[1] Regions to the right of the target verb corresponding to ancestor clauses are omitted in the case of partial parsing.

[2] With the unique exception of the exploration inside sibling PP constituents proposed by (Xue and Palmer, 2004).

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 193–196, Ann Arbor, June 2005.

lems to address multi-class classification.

AdaBoost binary classifiers are used for **labeling** test sequences in a left-to-right tagging scheme using a recurrent sliding window approach with information about the tag assigned to the preceding token. This tagging module ensures some basic constraints, e.g., BIO correct structure, arguments do not cross clause boundaries nor base chunk boundaries, `A0-A5` arguments not present in PropBank frames for a certain verb are not allowed, etc. We also tried beam search on top of the classifiers' predictions to find the sequence of labels with highest sentence-level probability (as a summation of individual predictions). But the results did not improve the basic greedy tagging.

Regarding **feature representation**, we used all input information sources, with the exception of verb senses and Collins' parser. We did not contribute with significantly original features. Instead, we borrowed most of them from the existing literature (Gildea and Jurafsky, 2002; Carreras et al., 2004; Xue and Palmer, 2004). Broadly speaking, we considered features belonging to four categories[3]:

(1) On the verb predicate:

- **Form**; **Lemma**; **POS tag**; **Chunk type** and **Type of verb phrase** in which verb is included: *single-word* or *multi-word*; **Verb voice**: *active*, *passive*, *copulative*, *infinitive*, or *progressive*; Binary flag indicating if the verb is a **start/end** of a clause.
- **Subcategorization**, i.e., the phrase structure rule expanding the verb parent node.

(2) On the focus constituent:

- **Type**; **Head**: extracted using common head-word rules; if the first element is a PP chunk, then the head of the first NP is extracted;
- **First and last words and POS tags** of the constituent.
- **POS sequence**: if it is less than 5 tags long; **2/3/4-grams** of the POS sequence.
- **Bag-of-words** of nouns, adjectives, and adverbs in the constituent.
- **TOP sequence**: sequence of types of the top-most syntactic elements in the constituent (if it is less than 5 elements long); in the case of full parsing this corresponds to the right-hand side of the rule expanding the constituent node; **2/3/4-grams** of the TOP sequence.
- **Governing category** as described in (Gildea and Jurafsky, 2002).
- **NamedEnt**, indicating if the constituent embeds or strictly-matches a named entity along with its type.
- **TMP**, indicating if the constituent embeds or strictly matches a temporal keyword (extracted from `AM-TMP` arguments of the training set).

(3) Context of the focus constituent:

- **Previous and following words and POS tags** of the constituent.
- The same features characterizing focus constituents are extracted for the **two previous and following tokens**, provided they are inside the clause boundaries of the codified region.

(4) Relation between predicate and constituent:

- **Relative position**; **Distance** in words and chunks; **Level of embedding** with respect to the constituent: in number of clauses.
- **Constituent path** as described in (Gildea and Jurafsky, 2002); All **3/4/5-grams** of path constituents beginning at the verb predicate or ending at the constituent.
- **Partial parsing path** as described in (Carreras et al., 2004); All **3/4/5-grams** of path elements beginning at the verb predicate or ending at the constituent.
- **Syntactic frame** as described by Xue and Palmer (2004)

2 Experimental Setting and Results

We trained the classification models using the complete training set (sections from 02 to 21). Once converted into one sequence per target predicate, the resulting set amounts 1,049,049 training examples in the PP_{UPC} model and 828,811 training examples in the FP_{CHA} model. The average number of labels per argument is 2.071 and 1.068, respectively. This fact makes "I" labels very rare in the FP_{CHA} model.

When running AdaBoost, we selected as weak rules decision trees of fixed depth 4 (i.e., each branch may represent a conjunction of at most 4 basic features) and trained a classification model per label for up to 2,000 rounds.

We applied some simplifications to keep training times and memory requirements inside admissible bounds. First, we discarded all the argument labels that occur very infrequently and trained only the 41 most frequent labels in the case of PP_{UPC} and the 35 most frequent in the case of FP_{CHA}. The remaining labels where joined in a new label "`other`" in training and converted into "`O`" whenever the SRL system assigns a "`other`" label during testing. Second, we performed a simple frequency filtering by discarding those features occurring less than 15 times in the training set. As an

[3] Features extracted from partial parsing and Named Entities are common to PP_{UPC} and FP_{CHA} models, while features coming from Charniak parse trees are implemented exclusively in the FP_{CHA} model.

exception, the frequency threshold for the features referring to the verb predicate was set to 3. The final number of features we worked with is 105,175 in the case of PP_{UPC} and 80,742 in the case of FP_{CHA}.

Training with these very large data and feature sets becomes an issue. Fortunately, we could split the computation among six machines in a Linux cluster. Using our current implementation combining Perl and C++ we could train the complete models in about 2 days using memory requirements between 1.5GB and 2GB. Testing with the ensembles of 2,000 decision trees per label is also not very efficient, though the resulting speed is admissible, e.g., the development set is tagged in about 30 minutes using a standard PC.

The overall results obtained by our individual PP_{UPC} and FP_{CHA} SRL systems are presented in table 1, with the best results in boldface. As expected, the FP_{CHA} system significantly outperformed the PP_{UPC} system, though the results of the later can be considered competitive. This fact is against the belief, expressed as one of the conclusions of the CoNLL-2004 shared task, that full-parsing systems are about 10 F_1 points over partial-parsing systems. In this case, we obtain a performance difference of 2.18 points in favor of FP_{CHA}.

Apart from resulting performance, there are additional advantages when using the FP_{CHA} approach. Due to the coarser granularity of sequence tokens, FP_{CHA} sequences are shorter. There are 21% less training examples and a much lower quantity of "I" tags to predict (the mapping between syntactic constituents and arguments is mostly one-to-one). As a consequence, FP_{CHA} classifiers train faster with less memory requirements, and achieve competitive results (near the optimal) with much less rounds of boosting. See figure 1. Also related to the token granularity, the number of completely correct outputs is 4.13 points higher in FP_{CHA}, showing that the resulting labelings are structurally better than those of PP_{UPC}.

Interestingly, the PP_{UPC} and FP_{CHA} systems make quite different argument predictions. For instance, FP_{CHA} is better at recognizing `A0` and `A1` arguments since parse constituents corresponding to these arguments tend to be mostly correct. Comparatively, PP_{UPC} is better at recognizing `A2-A4` arguments since they are further from the verb predicate

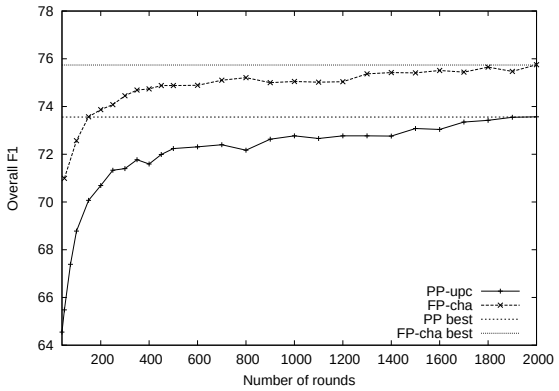


Figure 1: Overall F_1 performance of individual systems on the development set with respect to the number of learning rounds

	Perfect props	Precision	Recall	$F_{\beta=1}$
PP_{UPC}	47.38%	76.86%	70.55%	73.57
FP_{CHA}	**51.51**%	78.08%	73.54%	75.75
Combined	51.39%	**78.39**%	**75.53**%	**76.93**

Table 1: Overall results of the individual systems on the development set.

and tend to accumulate more parsing errors, while the fine granularity of the PP_{UPC} sequences still allow to capture them[4]. Another interesting observation is that the precision of both systems is much higher than the recall.

The previous two facts suggest that combining the outputs of the two systems may lead to a significant improvement. We experimented with a greedy combination scheme for joining the maximum number of arguments from both solutions in order to increase coverage and, hopefully, recall. It proceeds departing from an empty solution by: First, adding all the arguments from FP_{CHA} in which this method performs best; Second, adding all the arguments from PP_{UPC} in which this method performs best; and Third, making another loop through the two methods adding the arguments not considered in the first loop. At each step, we require that the added arguments do not overlap/embed with arguments in the current solution and also that they do not introduce repetitions of `A0-A5` arguments. The results on the

[4] As an example, the F_1 performance of PP_{UPC} on `A0` and `A2` arguments is 79.79 and 65.10, respectively. The performance of FP_{CHA} on the same arguments is 84.03 and 62.36.

	Precision	Recall	$F_{\beta=1}$
Development	78.39%	75.53%	76.93
Test WSJ	79.55%	76.45%	77.97
Test Brown	70.79%	64.35%	67.42
Test WSJ+Brown	78.44%	74.83%	76.59

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	79.55%	76.45%	77.97
A0	87.11%	86.28%	86.69
A1	79.60%	76.72%	78.13
A2	69.18%	67.75%	68.46
A3	76.38%	56.07%	64.67
A4	79.78%	69.61%	74.35
A5	0.00%	0.00%	0.00
AM-ADV	59.15%	52.37%	55.56
AM-CAU	73.68%	57.53%	64.62
AM-DIR	71.43%	35.29%	47.24
AM-DIS	77.14%	75.94%	76.54
AM-EXT	63.64%	43.75%	51.85
AM-LOC	62.74%	54.27%	58.20
AM-MNR	54.33%	52.91%	53.61
AM-MOD	96.16%	95.46%	95.81
AM-NEG	99.13%	98.70%	98.91
AM-PNC	53.49%	40.00%	45.77
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	77.68%	78.75%	78.21
R-A0	86.84%	88.39%	87.61
R-A1	75.32%	76.28%	75.80
R-A2	54.55%	37.50%	44.44
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	69.81%	71.15%	70.48
V	99.16%	99.16%	99.16

Table 2: Overall results (top) and detailed results on the WSJ test (bottom).

development set (presented in table 1) confirm our expectations, since a performance increase of 1.18 points over the best individual system was observed, mainly caused by recall improvement. The final system we presented at the shared task performs exactly this solution merging procedure. When applied on the WSJ test set, the combination scheme seems to generalize well, since an improvement is observed with respect to the development set. See the official results of our system, which are presented in table 2. Also from that table, it is worth noting that the F_1 performance drops by more than 9 points when tested on the Brown test set, indicating that the results obtained on the WSJ corpora do not generalize well to corpora with other genres. The study of the sources of this lower performance deserves further investigation, though we do not believe that it is attributable to the greedy combination scheme.

3 Conclusions

We have presented a simple SRL system submitted to the CoNLL-2005 shared task, which treats the SRL problem as a sequence tagging task (using a BIO tagging scheme). Given the simplicity of the approach, we believe that the results are very good and competitive compared to the state-of-the-art. We also provided a comparison between two SRL systems sharing the same architecture, but build on partial vs. full parsing, respectively. Although the full parsing approach obtains better results and has some implementation advantages, the partial parsing system shows also a quite competitive performance. The results on the development set differ in 2.18 points, but the outputs generated by the two systems are significantly different. The final system, which scored F_1=76.59 in the official test set, is a combination of both individual systems aiming at increasing coverage and recall.

Acknowledgements

This research has been partially supported by the European Commission (CHIL project, IP-506909). Jesús Giménez is a research fellow from the Spanish Ministry of Science and Technology (ALIADO project, TIC2002-04447-C02). We would like to thank also Xavier Carreras for providing us with many software components and Mihai Surdeanu for fruitful discussions on the problem and feature engineering.

References

X. Carreras, L. Màrquez, and G. Chrupała. 2004. Hierarchical recognition of propositional arguments with perceptrons. In *Proceedings of CoNLL-2004*.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

R. E. Schapire and Y. Singer. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3).

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Semantic Role Labeling Using Support Vector Machines

Tomohiro Mitsumori[1], **Masaki Murata**[2], **Yasushi Fukuda**[3]
Kouichi Doi[1], **and Hirohumi Doi**[1]

[1]Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama-cho, Ikoma-shi, Nara, 630-0101, Japan
{mitsumor,doy}@is.naist.jp, doi@cl-sciences.co.jp

[2]National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0289, Japan
murata@nict.go.jp

[3]Sony-Kihara Research Center Inc.
1-14-10 Higashigotanda, Shinagawa-ku, Tokyo, 141-0022, Japan
yasu@krc.sony.co.jp

Abstract

In this paper, we describe our systems for the CoNLL-2005 shared task. The aim of the task is semantic role labeling using a machine-learning algorithm. We apply the Support Vector Machines to the task. We added new features based on full parses and manually categorized words. We also report on system performance and what effect the newly added features had.

1 Introduction

The CoNLL-2005 shared task (Carreras and Màrquez, 2005) concerns the recognition of automatic semantic roles for the English language. Given a sentence, the task consists of analyzing the propositions expressed by various target verbs of the sentence. The semantic roles of constituents of the sentence are extracted for each target verb. There are semantic arguments such as Agent, Patient, and Instrument and also adjuncts such as Locative and Temporal. We performed the semantic role labeling using Support Vector Machines (SVMs). Systems that used SVMs achieved good performance in the CoNLL-2004 shared task, and we added data on full parses to it. We prepared a feature used by the full parses, and we also categorized words that appeared in the training set and added them as features. Here, we report on systems for automatically labeling semantic roles in a closed challenge in the CoNLL-2005 shared task.

This paper is arranged as follows. Section 2 describes the SVMs. Our system is described Section 3, where we also describe methods of data representation, feature coding, and the parameters of SVMs. The experimental results and conclusion are presented in Sections 4 and 5.

2 Support Vector Machines

SVMs are one of the binary classifiers based on the maximum margin strategy introduced by Vapnik (Vapnik, 1995). This algorithm has achieved good performance in many classification tasks, e.g. named entity recognition and document classification. There are some advantages to SVMs in that (i) they have high generalization performance independent of the dimensions of the feature vectors and (ii) learning with a combination of multiple features is possible by using the polynomial kernel function (Yamada and Matsumoto, 2003). SVMs were used in the CoNLL-2004 shred task and achieved good performance (Hacioglu et al., 2004) (Kyung-Mi Park and Rim, 2004). We used YamCha (Yet Another Multipurpose Chunk Annotator)[1] (Kudo and Matsumoto, 2001), which is a general purpose SVM-based chunker. We also used TinySVM[2] as a package for SVMs.

3 System Description

3.1 Data Representation

We changed the representation of original data according to Hacioglu et al. (Hacioglu et al., 2004) in our system.

[1]http://chasen.org/~ taku/software/yamcha/
[2]http://chasen.org/~ taku/software/TinySVM/

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 197–200, Ann Arbor, June 2005.

- Bracketed representation of roles was converted into IOB2 representation (Ramhsaw and Marcus, 1995) (Sang and Veenstra, 1999).

- Word-by-word was changed to the phrase-by-phrase method (Hacioglu et al., 2004).

Word tokens were collapsed into base phrase (BP) tokens. The BP headwords were rightmost words. Verb phrases were not collapsed because some included more the one predicate.

3.2 Feature Coding

We prepared the training and development set by using files corresponding to: words, predicated partial parsing (part-of-speech, base chunks), predicate full parsing trees (Charniak models), and named entities. We will describe feature extraction according to Fig. 1. Figure 1 shows an example of an annotated sentence.

1st Words (Bag of Words): All words appearing in the training data.

2nd Part of Speech (POS) Tags

3rd Base Phrase Tags: Partial parses (chunks + clauses) predicted with UPC processors.

4th Named Entities

5th Token Depth : This means the degree of depth from a predicate (see Fig. 2). We used full parses predicted by the Charniak parser. In this figure, the depth of paid, which is a predicate, is zero and the depth of April is -2.

6th Words of Predicate

7th Position of Tokens: The position of the current word from the predicate. This has three value of "before", "after", and "-" (for the predicate).

8th Phrase Distance on Flat Path: This means the distance from the current token to the predicate as a number of the phrase on flat path. For example, the phrase distance of "April" is 4, because two "NP" and one "PP" exist from "paid"(predicate) to "April" (see 3rd column in Fig.1).

Table 1: Five most frequently categorized BP headwords appearing in training set.

Class	Examples
Person	he, I, people, investors, we
Organization	company, Corp., Inc., companies, group
Time	year, years, time, yesterday, months
Location	Francisco, York, California, city, America
Number	%, million, billion, number, quarter
Money	price, prices, cents, money, dollars

9th Flat Path: This means the path from the current word to the predicate as a chain of the phrases. The chain begins from the BP of the current word to the BP of the predicate.

10th Semantic Class : We collected the most frequently occurring 1,000 BP headwords appearing in the training set and tried to manually classified. The five classes (person, organization, time, location, number and money) were relatively easy to classify. In the 1,000 words, the 343 words could be classified into the five classes. Remainder could not be classified. The details are listed in Table 1.

Preceding class: The class (e.g. B-A0 or I-A1) of the token(s) preceding the current token. The number of preceding tokens is dependent on the window size. In this paper, the left context considered is two.

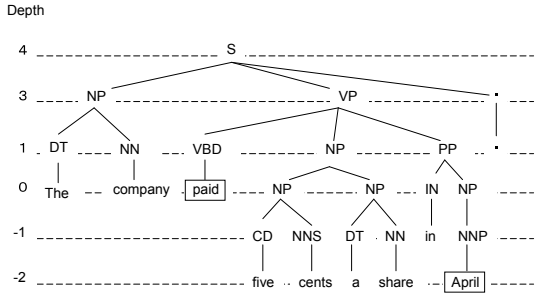


Figure 2: Parsing results obtained with Charniak parser and token depth.

3.3 Machine learning with YamCha

YamCha (Kudo and Matsumoto, 2001) is a general purpose SVM-based chunker. After inputting the training and test data, YamCha converts them for

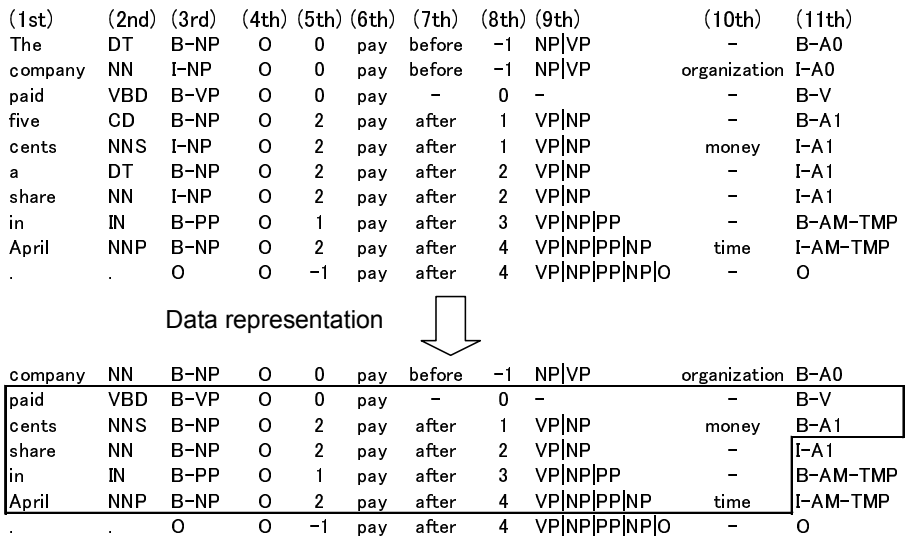


Figure 1: Example annotated sentence. Input features are words (1st), POS tags (2nd), base phrase chunks (3rd), named entities (4th), token depth (5th), predicate (6th), position of tokens (7th), phrase distance (8th), flat paths (9th), semantic classes (10th), argument classes (11th).

the SVM. The YamCha format for an example sentence is shown in Fig. 1. Input features are written in each column as words (1st), POS tags (2nd), base phrase chunks (3rd), named entities (4th), token depth (5th), predicate (6th), the position of tokens (7th), the phrase distance (8th), flat paths (9th), semantic classes (10th), and argument classes (11th). The boxed area in Fig. 1 shows the elements of feature vectors for the current word, in this case "share". The information from the two preceding and two following tokens is used for each vector.

3.4 Parameters of SVM

- Degree of polynomial kernel (natural number): We can only use a polynomial kernel in YamCha. In this paper, we adopted the degree of two.

- Range of window (integer): The SVM can use the information on tokens surrounding the token of interest as illustrated in Fig. 1. In this paper, we adopted the range from the left two tokens to the right two tokens.

- Method of solving a multi-class problem: We adopted the one-vs.-rest method. The BIO class is learned as (B vs. other), (I vs. other), and (O vs. other).

- Cost of constraint violation (floating number): There is a trade-off between the training error and the soft margin for the hyper plane. We adopted a default value (1.0).

4 Results

4.1 Data

The data provided for the shared task consisted of sections from the Wall Street Journal (WSJ) part of Penn TreeBank II. The training set was WSJ Sections 02-21, the development set was Section 24, and the test set was Section 23 with the addition of fresh sentences. Our experiments were carried out using Sections 15-18 for the training set, because the entire file was too large to learn.

4.2 Experiments

Our final results for the CoNLL-2005 shared task are listed in Table 2. Our system achieved 74.15% precision, 68.25% recall and 71.08 $F_{\beta=1}$ on the overall results of Test WSJ. Table 3 lists the effects of the token-depth and semantic-class features. The token-depth feature had a larger effect than that for the semantic class.

	Precision	Recall	$F_{\beta=1}$
Development	71.68%	64.93%	68.14
Test WSJ	74.15%	68.25%	71.08
Test Brown	63.24%	54.20%	58.37
Test WSJ+Brown	72.77%	66.37%	69.43

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	74.15%	68.25%	71.08
A0	81.38%	76.93%	79.09
A1	73.16%	70.87%	72.00
A2	64.53%	59.01%	61.65
A3	61.16%	42.77%	50.34
A4	68.18%	58.82%	63.16
A5	100.00%	80.00%	88.89
AM-ADV	55.09%	43.87%	48.84
AM-CAU	60.00%	28.77%	38.89
AM-DIR	45.10%	27.06%	33.82
AM-DIS	72.70%	69.06%	70.83
AM-EXT	70.59%	37.50%	48.98
AM-LOC	55.62%	50.41%	52.89
AM-MNR	51.40%	42.73%	46.67
AM-MOD	97.04%	95.28%	96.15
AM-NEG	96.92%	95.65%	96.28
AM-PNC	56.00%	36.52%	44.21
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	73.39%	62.93%	67.76
R-A0	81.31%	71.88%	76.30
R-A1	59.69%	49.36%	54.04
R-A2	60.00%	18.75%	28.57
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	28.57%	42.86
R-AM-MNR	100.00%	16.67%	28.57
R-AM-TMP	72.34%	65.38%	68.69
V	97.55%	96.05%	96.80

Table 2: Overall results (top) and detailed results on the WSJ test (bottom).

5 Conclusion

This paper reported on semantic role labeling using SVMs. Systems that used SVMs achieved good performance in the CoNLL-2004 shared task, and we added data on full parses to it. We applied a token-depth feature to SVM learning and it had a large effect. We also added a semantic-class feature and it had a small effect. Some classes were similar to the named entities, e.g., the PERSON or LOCATION of the named entities. Our semantic class feature also included not only proper names but also common words. For example, "he" and "she" also included the PERSON semantic class. Furthermore, we added a time, number, and money class. The semantic class feature was manually categorized on the most frequently occurring 1,000 words in the training set. More effort of the categorization may improve the performance of our system.

Table 3: Effects Token Depth (TD) and Semantic Class (SC) had on feature development set.

	Precision	Recall	$F_{\beta=1}$
Without DF and SC	68.07%	59.71%	63.62
With DF	71.36%	64.13%	67.55
With DF and SC	71.68%	64.93%	68.14

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling . In *Proceedings of CoNLL-2005*.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H. Martin, and Daniel Jurafskey. 2004. Semantic Role Labeling by Tagging Syntactic Chunks. In *Proceedings of Conference on Computational Natural Language Learning (CoNLL-2004) Shared Task on Semantic Role Labeling*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proceedings of Second Meeting of North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 192–199.

Young-Sook Hwang Kyung-Mi Park and Hae-Chang Rim. 2004. Semantic Role Labeling by Tagging Syntactic Chunks. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-2004) Shared Task on Semantic Role Labeling*.

Lance E. Ramhsaw and Mitchell P. Marcus. 1995. Text Chunking Using Transformation Based Learning . In *Proceedings of the 3rd ACL Workshop on Very Large Corpora*, pages 82–94.

Erik F. T. J. Sang and John Veenstra. 1999. Representing Text Chunks. In *Proceedings of EACL !99*, pages 173–179.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with Support Vector Machines . In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Hierarchical Semantic Role Labeling

Alessandro Moschitti[◇] **Ana-Maria Giuglea**[◇] **Bonaventura Coppola**[†‡] **Roberto Basili**[◇]
moschitti@info.uniroma2.it ana-maria.giuglea@topex.ro coppolab@itc.it basili@info.uniroma2.it

[◇] DISP - University of Rome "Tor Vergata", Rome, Italy
[†] ITC-Irst, [‡] DIT - University of Trento, Povo, Trento, Italy

Abstract

We present a four-step hierarchical SRL strategy which generalizes the classical two-level approach (boundary detection and classification). To achieve this, we have split the classification step by grouping together roles which share linguistic properties (e.g. Core Roles versus Adjuncts). The results show that the non-optimized hierarchical approach is computationally more efficient than the traditional systems and it preserves their accuracy.

1 Introduction

For accomplishing the CoNLL 2005 Shared Task on Semantic Role Labeling (Carreras and Màrquez, 2005), we capitalized on our experience on the semantic shallow parsing by extending our system, widely experimented on PropBank and FrameNet (Giuglea and Moschitti, 2004) data, with a two-step boundary detection and a hierarchical argument classification strategy.

Currently, the system can work in both basic and enhanced configuration. Given the parse tree of an input sentence, the basic system applies (1) a boundary classifier to select the nodes associated with correct arguments and (2) a multi-class labeler to assign the role type. For such models, we used some of the linear (e.g. (Gildea and Jurasfky, 2002; Pradhan et al., 2005)) and structural (Moschitti, 2004) features developed in previous studies.

In the enhanced configuration, the boundary annotation is subdivided in two steps: a first pass in which we label argument boundary and a second pass in which we apply a simple heuristic to eliminate the argument overlaps. We have also tried some strategies to learn such heuristics automatically. In order to do this we used a tree kernel to classify the subtrees associated with correct predicate argument structures (see (Moschitti et al., 2005)). The rationale behind such an attempt was to exploit the correlation among potential arguments.

Also, the role labeler is divided into two steps: (1) we assign to the arguments one out of four possible class labels: *Core Roles*, *Adjuncts*, *Continuation Arguments* and *Co-referring Arguments*, and (2) in each of the above class we apply the set of its specific classifiers, e.g. A0,..,A5 within the Core Role class. As such grouping is relatively new, the traditional features may not be sufficient to characterize each class. Thus, to generate a large set of features automatically, we again applied tree kernels.

Since our SRL system exploits the PropBank formalism for internal data representation, we developed ad-hoc procedures to convert back and forth to the CoNLL Shared Task format. This conversion step gave us useful information about the amount and the nature of the parsing errors. Also, we could measure the frequency of the mismatches between syntax and role annotation.

In the remainder of this paper, Section 2 describes the basic system configuration whereas Section 3 illustrates its enhanced properties and the hierarchical structure. Section 4 describes the experimental setting and the results. Finally, Section 5 summarizes

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 201–204, Ann Arbor, June 2005.

our conclusions.

2 The Basic Semantic Role Labeler

In the last years, several machine learning approaches have been developed for automatic role labeling, e.g. (Gildea and Jurasfky, 2002; Pradhan et al., 2005). Their common characteristic is the adoption of flat feature representations for predicate-argument structures. Our basic system is similar to the one proposed in (Pradhan et al., 2005) and it is described hereafter.

We divided the predicate argument labeling in two subtasks: (a) the detection of the arguments related to a target, i.e. all the compounding words of such argument, and (b) the classification of the argument type, e.g. A0 or AM. To learn both tasks we used the following algorithm:

1. Given a sentence from the *training-set*, generate a full syntactic parse-tree;
2. Let $\mathcal{P}$ and $\mathcal{A}$ be respectively the set of predicates and the set of parse-tree nodes (i.e. the potential arguments);
3. For each pair $<p, a> \in \mathcal{P} \times \mathcal{A}$:
   - extract the feature representation set, $F_{p,a}$;
   - if the subtree rooted in a covers exactly the words of one argument of p, put $F_{p,a}$ in T^+ (positive examples), otherwise put it in T^- (negative examples).

We trained the SVM boundary classifier on T^+ and T^- sets and the role labeler i on the T_i^+, i.e. its positive examples and T_i^-, i.e. its negative examples, where $T^+ = T_i^+ \cup T_i^-$, according to the ONE-vs.-ALL scheme. To implement the multi-class classifiers we select the argument associated with the maximum among the SVM scores.

To represent the $F_{p,a}$ pairs we used the following features:
- the *Phrase Type*, *Predicate Word*, *Head Word*, *Governing Category*, *Position* and *Voice* defined in (Gildea and Jurasfky, 2002);
- the *Partial Path*, *Compressed Path*, *No Direction Path*, *Constituent Tree Distance*, *Head Word POS*, *First and Last Word/POS in Constituent*, *SubCategorization* and *Head Word of Prepositional Phrases* proposed in (Pradhan et al., 2005); and
- the *Syntactic Frame* designed in (Xue and Palmer, 2004).

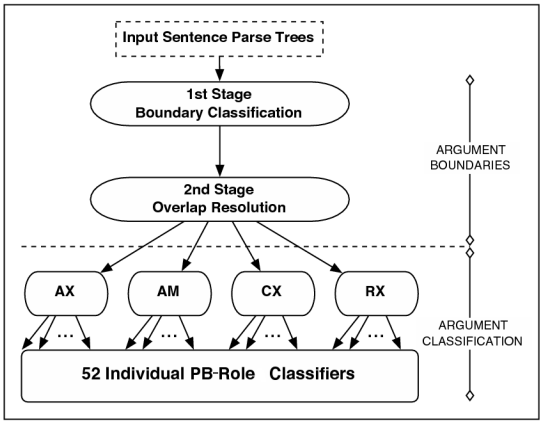


Figure 1: Architecture of the Hierarchical Semantic Role Labeler.

3 Hierarchical Semantic Role Labeler

Having two phases for argument labeling provides two main advantages: (1) the efficiency is increased as the negative boundary examples, which are almost all parse-tree nodes, are used with one classifier only (i.e. the boundary classifier), and (2) as arguments share common features that do not occur in the non-arguments, a preliminary classification between arguments and non-arguments advantages the boundary detection of roles with fewer training examples (e.g. A4). Moreover, it may be simpler to classify the type of roles when the not-argument nodes are absent.

Following this idea, we generalized the above two level strategy to a four-step role labeling by grouping together the arguments sharing similar properties. Figure 1, shows the hierarchy employed for argument classification:

During the first phase, we select the parse tree nodes which are likely predicate arguments. An SVM with moderately high recall is applied for such purpose.

In the second phase, a simple heuristic which selects *non-overlapping* nodes from those derived in the previous step is applied. Two nodes n_1 and n_2 do not overlap if n_1 is not ancestor of n_2 and viceversa. Our heuristic simply eliminates the nodes that cause the highest number of overlaps. We have also studied how to train an overlap resolver by means of tree kernels; the promising approach and results can be found in (Moschitti et al., 2005).

In the third phase, we classify the detected arguments in the following four classes: AX, i.e. *Core*

Arguments, AM, i.e. *Adjuncts*, CX, i.e. *Continuation Arguments* and RX, i.e. the *Co-referring Arguments*. The above classification relies on linguistic reasons. For example *Core arguments* class contains the arguments specific to the verb frames while *Adjunct Arguments* class contains arguments that are shared across all verb frames.

In the fourth phase, we classify the members within the classes of the previous level, e.g. `A0` vs. `A1`, ..., `A5`.

4 The Experiments

We experimented our approach with the CoNLL 2005 Shared Task standard dataset, i.e. the PennTree Bank, where sections from 02 to 21 are used as training set, Section 24 as development set (Dev) and Section 23 as the test set (WSJ). Additionally, the Brown corpus' sentences were also used as the test set (Brown). As input for our feature extractor we used only the Charniak's parses with their POSs.

The evaluations were carried out with the SVM-light-TK software (Moschitti, 2004) available at `http://ai-nlp.info.uniroma2.it/moschitti/` which encodes the tree kernels in the SVM-light software (Joachims, 1999). We used the default polynomial kernel (degree=3) for the linear feature representations and the tree kernels for the structural feature processing.

As our feature extraction module was designed to work on the PropBank project annotation format (i.e. the *prop.txt* index file), we needed to generate it from the CoNLL data. Each PropBank annotation refers to a parse tree node which exactly covers the target argument but when using automatic parses such node may not exist. For example, on the CoNLL Charniak's parses, (sections 02-21 and 24), we discovered that this problem affects 10,293 out of the 241,121 arguments (4.3%) and 9,741 sentences out of 87,257 (11.5%). We have found out that most of the errors are due to wrong parsing attachments. This observation suggests that the capability of discriminating between correct and incorrect parse trees is a key issue in the boundary detection phase and it must be properly taken into account.

4.1 Basic System Evaluation

For the boundary classifier we used a SVM with the polynomial kernel of degree 3. We set the regularization parameter, `c`, to 1 and the cost factor, `j` to 7 (to have a slightly higher recall). To reduce the learning time, we applied a simple heuristic which removes the nodes covering the target predicate node. From the initial 4,683,777 nodes (of sections 02-21), the heuristic removed 1,503,100 nodes with a loss of 2.6% of the total arguments. However, as we started the experiments in late, we used only the 992,819 nodes from the sections 02-08. The classifier took about two days and half to converge on a 64 bits machine (2.4 GHz and 4Gb Ram).

The multiclassifier was built with 52 binary argument classifiers. Their training on all arguments from sec 02-21, (i.e. 242,957), required about a half day on a machine with 8 processors (32 bits, 1.7 GHz and overll 4Gb Ram).

We run the role multiclassifier on the output of the boundary classifier. The results on the Dev, WSJ and Brown test data are shown in Table 1. Note that, the overlapping nodes cause the generation of overlapping constituents in the sentence annotation. This prevents us to use the CoNLL evaluator. Thus, we used the overlap resolution algorithm also for the basic system.

4.2 Hierarchical Role Labeling Evaluation

As the first two phases of the hierarchical labeler are identical to the basic system, we focused on the last two phases. We carried out our studies over the Gold Standard boundaries in the presence of arguments that do not have a *perfect-covering* node in the Charniak trees.

To accomplish the third phase, we re-organized the flat arguments into the AX, AM, CX and RX classes and we built a single multi-classifier. For the fourth phase, we built a multi-classifier for each of the above classes: only the examples related to the target class were used, e.g. the AX mutliclassifier was designed with the A0,..,A5 ONE-vs-ALL binary classifiers.

In rows 2 and 3, Table 2 shows the numbers of training and development set instances. Row 4 contains the F_1 of the binary classifiers of the third phase whereas Row 5 reports the F_1 of the resulting multi-classifier. Row 6 presents the F_1s of the multi-classifiers of the fourth phase.

Row 7 illustrates the F_1 measure of the fourth phase classifier applied to the third phase output. Fi-

	Precision	Recall	$F_{\beta=1}$
Development	74.95%	73.10%	74.01
Test WSJ	76.55%	75.24%	75.89
Test Brown	65.92%	61.83%	63.81
Test WSJ+Brown	75.19%	73.45%	74.31

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	76.55%	75.24%	75.89
A0	81.05%	84.37%	82.67
A1	77.21%	74.12%	75.63
A2	67.02%	68.11%	67.56
A3	69.63%	54.34%	61.04
A4	74.75%	72.55%	73.63
A5	100.00%	40.00%	57.14
AM-ADV	55.23%	55.34%	55.28
AM-CAU	66.07%	50.68%	57.36
AM-DIR	50.62%	48.24%	49.40
AM-DIS	77.71%	78.44%	78.07
AM-EXT	68.00%	53.12%	59.65
AM-LOC	59.02%	63.09%	60.99
AM-MNR	67.67%	52.33%	59.02
AM-MOD	98.65%	92.56%	95.51
AM-NEG	97.37%	96.52%	96.94
AM-PNC	42.28%	45.22%	43.70
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	81.90%	74.52%	78.03
R-A0	79.50%	84.82%	82.07
R-A1	62.23%	75.00%	68.02
R-A2	100.00%	31.25%	47.62
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	50.00%	66.67
R-AM-EXT	100.00%	100.00%	100.00
R-AM-LOC	85.71%	85.71%	85.71
R-AM-MNR	22.22%	33.33%	26.67
R-AM-TMP	67.69%	84.62%	75.21
V	97.34%	97.30%	97.32

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

nally, in Row 8, we report the F_1 of the basic system on the gold boundary nodes. We note that the basic system shows a slightly higher F_1 but is less computational efficient than the hierarchical approach.

5 Final Remarks

In this paper we analyzed the impact of a hierarchical categorization on the semantic role labeling task. The results show that such approach produces an accuracy similar to the flat systems with a higher efficiency. Moreover, some preliminary experiments show that each node of the hierarchy requires different features to optimize the associated multiclassifier. For example, we found that the SCF tree kernel (Moschitti, 2004) improves the AX multiclassifier

	AX	AM	CX	RX
# train. examples	172,457	59,473	2,954	7,928
# devel. examples	5,930	2,132	105	284
Phase III: binary class.	97.29	97.35	70.86	93.15
Phase III	95.99			
Phase IV	92.50	85.88	91.43	91.55
Phase III & IV	88.15			
Basic System	88.61			

Table 2: Hierarchical Semantic Role Labeler Results

whereas the PAF tree kernel seems more suited for the classification within the other classes, e.g. AM.

Future work on the optimization of each phase is needed to study the potential accuracy limits of the proposed hierarchical approach.

Acknowledgements

We wish to thank Daniele Pighin for his valuable support in the development of the SRL system.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *proceedings of CoNLL'05*.

Daniel Gildea and Daniel Jurasfky. 2002. Automatic labeling of semantic roles. *Computational Linguistic*.

Ana-Maria Giuglea and Alessandro Moschitti. 2004. Knowledge Discovering using FrameNet, VerbNet and PropBank. In *proceedings of the Workshop on Ontology and Knowledge Discovering at ECML'04, Pisa, Italy*.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*.

Alessandro Moschitti, Bonaventura Coppola, Daniele Pighin, and Roberto Basili. 2005. Engineering of syntactic features for shallow semantic parsing. In *proceedings of the Feature Engineering Workshop at ACL'05*, Ann Arbor, USA.

Alessandro Moschitti. 2004. A study on convolution kernel for shallow semantic parsing. In *proceedings of ACL-2004*, Barcelona, Spain.

Sameer Pradhan, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Support vector learning for semantic argument classification. *to appear in Machine Learning Journal*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP'04*, Barcelona, Spain.

Semantic Role Labeling using libSVM

Necati Ercan Ozgencil
Center for Natural Language Processing
School of Engineering and Computer Science
Syracuse University
neozgenc@ecs.syr.edu

Nancy McCracken
Center for Natural Language Processing
School of Information Studies
Syracuse University
njm@ecs.syr.edu

Abstract

We describe a system for the CoNLL-2005 shared task of Semantic Role Labeling. The system implements a two-layer architecture to first identify the arguments and then to label them for each predicate. The components are implemented as SVM classifiers using libSVM. Features were adapted and tuned for the system, including a reduced set for the identifier classifier. Experiments were conducted to find kernel parameters for the Radial Basis Function (RBF) kernel. An algorithm was defined to combine the results of the argument labeling classifier according to the constraints of the argument labeling problem.

1 Introduction and Strategy

The Semantic Role Labeling (SRL) problem has been the topic of the both the CoNLL-2004 and the CoNLL-2005 Shared Tasks (Carreras and Màrquez, 2005). The SRL system described here depends on a full syntactic parse from the Charniak parser, and investigates aspects of using Support Vector Machines (SVMs) as the machine learning technique for the SRL problem, using the libSVM package.

In common with many other systems, this system uses the two-level strategy of first identifying which phrases can be arguments to predicates in general, and then labeling the arguments according to that predicate. The argument identification phase is a binary classifier that decides whether each constituent in the full syntax tree of the sentence is a potential argument. These potential arguments are passed into the argument labeling classifier, which uses binary classifiers for each label to decide if that label should be given to that argument. A post-processing phase picks the best labeling that satisfies the constraints of labeling the predicate arguments.

For overall classification strategy and for suggestions of features, we are indebted to the work of Pradhan et al (2005) and to the work of many authors in both the CoNLL-2004 shared task and the similar semantic roles task of Senseval-3. We used the results of their experiments with features, and worked primarily on features for the identifying classifier and with the constraint satisfaction problem on the final argument output.

2 System Description

2.1 Input Data

In this system, we chose to use full syntax trees from the Charniak parser, as the constituents of those trees more accurately represented argument phrases in the training data at the time of the data release. Within each sentence, we first map the predicate to a constituent in the syntax tree. In the cases that the predicate is not represented by a constituent, we found that these were verb phrases of length two or more, where the first word was the main verb (carry out, gotten away, served up, etc.). In these cases, we used the first word constituent as the representation of the predicate, for purposes of computing other features that depended on a relative position in the syntax tree.

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 205–208, Ann Arbor, June 2005.

We next identify every constituent in the tree as a potential argument, and label the training data accordingly. Although approximately 97% of the arguments in the training data directly matched constituents in the Charniak tree, only 91.3% of the arguments in the development set match constituents. Examination of the sentences with incorrect parses show that almost all of these are due to some form of incorrect attachment, e.g. prepositional attachment, of the parser. Heuristics can be derived to correct constituents with quotes, but this only affected a small fraction of a percent of the incorrect arguments. Experiments with corrections to the punctuation in the Collins parses were also unsuccessful in identifying additional constituents. Our recall results on the development directory are bounded by the 91.3% alignment figure.

We also did not use the the partial syntax, named entities or the verb senses in the development data.

2.2 Learning Components: SVM classifiers

For our system, we chose to use libSVM, an open source SVM package (Chang and Lin, 2001).

In the SRL problem, the features are nominal, and we followed the standard practice of representing a nominal feature with n discrete values as n binary features. Many of the features in the SRL problem can take on a large number of values, for example, the head word of a constituent may take on as many values as there are different words present in the training set, and these large number of features can cause substantial performance issues.

The libSVM package has several kernel functions available, and we chose to use the radial basis functions (RBF). For the argument labeling problem, we used the binary classifiers in libSVM, with probability estimates of how well the label fits the distribution. These are normally combined using the "one-against-one" approach into a multi-class classifier. Instead, we combined the binary classifiers in our own post-processing phase to get a labeling satisfying the constraints of the problem.

2.3 The Identifier Classifier Features

One aspect of our work was to use fewer features for the identifier classifier than the basic feature set from (Gildea and Jurafsky, 2002). The intuition behind the reduction is that whether a constituent in the tree is an argument depends primarily on the structure and is independent of the lexical items of the predicate and headword. This reduced feature set is:

Phrase Type: The phrase label of the argument.

Position: Whether the phrase is before or after the predicate.

Voice: Whether the predicate is in active or passive voice. Passive voice is recognized if a past participle verb is preceded by a form of the verb "be" within 3 words.

Sub-categorization: The phrase labels of the children of the predicate's parent in the syntax tree.

Short Path: The path from the parent of the argument position in the syntax tree to the parent of the predicate.

The first four features are standard, and the short path feature is defined as a shorter version of the standard path feature that does not use the argument phrase type on one end of the path, nor the predicate type on the other end.

The use of this reduced set of features was confirmed experimentally by comparing the effect of this reduced feature set on the F-measure of the identifier classifier, compared to feature sets that also added the predicate, the head word and the path features, as normally defined.

	Reduced	+ Pred	+ Head	+ Path
F-measure	81.51	81.31	72.60	81.19

Table 1: Additional features reduce F-measure for the identifier classifier.

2.4 Using the Identifier Classifier for Training and Testing

Theoretically, the input for training the identifier classifier is that, for each predicate, all constituents in the syntax tree are training instances, labeled true if it is any argument of that predicate, and false otherwise. However, this leads to too many negative (false) instances for the training. To correct this, we experimented with two filters for negative instances. The first filter is simply a random filter; we randomly select a percentage of arguments for each argument label. Experiments with the percentage showed that 30% yielded the best F-measure for the identifier classifier.

The second filter is based on phrase labels from the syntax tree. The intent of this filter was to remove one word constituents of a phrase type that was never used. We selected only those phrase

labels whose frequency in the training was higher than a threshold. Experiments showed that the best threshold was 0.01, which resulted in approximately 86% negative training instances.

However, in the final experimentation, comparison of these two filters showed that the random filter was best for F-measure results of the identifier classifier.

The final set of experiments for the identifier classifier was to fine tune the RBF kernel training parameters, C and gamma. Although we followed the standard grid strategy of finding the best parameters, unlike the built-in grid program of libSVM with its accuracy measure, we judged the results based on the more standard F-measure of the classifier. The final values are that C = 2 and gamma = 0.125.

The final result of the identifier classifier trained on the first 10 directories of the training set is:

Precision: 78.27% Recall: 89.01%
(F-measure: 83.47)

Training on more directories did not substantially improve these precision and recall figures.

2.5 Labeling Classifier Features

The following is a list of the features used in the labeling classifiers.

Predicate: The predicate lemma from the training file.

Path: The syntactic path through the parse tree from the argument constituent to the predicate.

Head Word: The head word of the argument constituent, calculated in the standard way, but also stemmed. Applying stemming reduces the number of unique values of this feature substantially, 62% in one directory of training data.

Phrase Type, Position, Voice, and Subcategorization: as in the identifier classifier.

In addition, we experimented with the following features, but did not find that they increased the labeling classifier scores.

Head Word POS: the part of speech tag of the head word of the argument constituent.

Temporal Cue Words: These words were compiled by hand from ArgM-TMP phrases in the training data.

Governing Category: The phrase label of the parent of the argument.

Grammatical Rule: The generalization of the subcategorization feature to show the phrase labels of the children of the node that is the lowest parent of all arguments of the predicate.

In the case of the temporal cue words, we noticed that using our definition of this feature increased the number of false positives for the `ARGM-TMP` label; we guess that our temporal cue words included too many words that occured in other labels. Due to lack of time, we were not able to more fully pursue these features.

2.6 Using the Labeling Classifier for Training and Testing

Our strategy for using the labeling classifier is that in the testing, we pass only those arguments to the labeling classifier that have been marked as true by the identifier classifier. Therefore, for training the labeling classifier, instances were constituents that were given argument labels in the training set, i.e. there were no "null" training examples.

For the labeling classifier, we also found the best parameters for the RBF kernel of the classifier. For this, we used the grid program of libSVM that uses the multi-class classifier, using the accuracy measure to tune the parameters, since this combines the precision of the binary classifiers for each label. The final values are that C = 0.5 and gamma = 0.5.

In order to show the contribution of the labeling classifier to the entire system, a final test was done on the development set, but passing it the correct arguments. We tested this with a labeling classifier trained on 10 directories and one trained on 20 directories, showing the final F-measure:

10 directories:	83.27
20 directories:	84.51

2.7 Post-processing the classifier labels

The final part of our system was to use the results of the binary classifiers for each argument label to produce a final labeling subject to the labeling constraints.

For each predicate, the constraints are: two constituents cannot have the same argument label, a constituent cannot have more than one label, if two constituents have (different) labels, they cannot have any overlap, and finally, no argument can overlap the predicate.

	Precision	Recall	$F_{\beta=1}$
Development	73.57%	71.87%	72.71
Test WSJ	74.66%	74.21%	74.44
Test Brown	65.52%	62.93%	64.20
Test WSJ+Brown	73.48%	72.70%	73.09

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	74.66%	74.21%	74.44
A0	83.59%	85.07%	84.32
A1	77.00%	74.35%	75.65
A2	66.97%	66.85%	66.91
A3	66.88%	60.69%	63.64
A4	77.66%	71.57%	74.49
A5	80.00%	80.00%	80.00
AM-ADV	55.13%	50.99%	52.98
AM-CAU	52.17%	49.32%	50.70
AM-DIR	27.43%	56.47%	36.92
AM-DIS	73.04%	72.81%	72.93
AM-EXT	57.69%	46.88%	51.72
AM-LOC	50.00%	49.59%	49.79
AM-MNR	54.00%	54.94%	54.47
AM-MOD	92.02%	94.19%	93.09
AM-NEG	96.05%	95.22%	95.63
AM-PNC	35.07%	40.87%	37.75
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	68.69%	63.57%	66.03
R-A0	77.61%	89.73%	83.23
R-A1	71.95%	75.64%	73.75
R-A2	87.50%	43.75%	58.33
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	50.00%	66.67
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	66.67%	85.71%	75.00
R-AM-MNR	8.33%	16.67%	11.11
R-AM-TMP	66.67%	88.46%	76.03
V	97.32%	97.32%	97.32

Table 2: Overall results (top) and detailed results on the WSJ test (bottom).

To achieve these constraints, we used the probabilities produced by libSVM for each of the binary argument label classifiers. We produced a constraint satisfaction module that uses a greedy algorithm that uses probabilities from the matrix of potential labeling for each constituent and label. The algorithm iteratively chooses a label for a node with the highest probability and removes any potential labeling that would violate constraints with that chosen label. It continues to choose labels for nodes until all probabilities in the matrix are lower than a threshold, determined by experiments to be .3. In the future, it is our intent to replace this greedy algorithm with a dynamic optimization algorithm.

3 Experimental Results

3.1 Final System and Results

The final system used an identifier classifier trained on (the first) 10 directories, in approximately 7 hours, and a labeling classifier trained on 20 directories, in approximately 23 hours. Testing took approximately 3.3 seconds per sentence.

As a further test of the final system, we trained both the identifier classifier and the labeling classifier on the first 10 directories and used the second 10 directories as development tests. Here are some of the results, showing the alignment and F-measure on each directory, compared to 24.

Directory:	12	14	16	18	20	24
Alignment	95.7	96.1	95.9	96.5	95.9	91.3
F-measure	80.4	79.6	79.0	80.5	79.7	71.1

Table 3: Using additional directories for testing

Finally, we note that we did not correctly anticipate the final notation for the predicates in the test set for two word verbs. Our system assumed that two word verbs would be given a start and an end, whereas the test set gives just the one word predicate.

References

Xavier Carreras and Lluìs Màrquez, 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling, Proceedings of CoNLL-2005.

Chih-Chung Chang and Chih-Jen Lin, 2001. LIBSVM : a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Daniel Gildea and Daniel Jurafsky, 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics* 28(3):245-288.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, and Daniel Jurafsky, 2005. Support Vector Learning for Semantic Argument Classification, To appear in *Machine Learning* journal, Special issue on Speech and Natural Language Processing.

Maximum Entropy based Semantic Role Labeling

Kyung-Mi Park and **Hae-Chang Rim**
Department of Computer Science & Engineering, Korea University
5-ka, Anam-dong, SeongBuk-gu, SEOUL, 136-701, KOREA
{kmpark, rim}@nlp.korea.ac.kr

1 Introduction

The semantic role labeling (SRL) refers to finding the semantic relation (e.g. *Agent*, *Patient*, etc.) between a predicate and syntactic constituents in the sentences. Especially, with the argument information of the predicate, we can derive the predicate-argument structures, which are useful for the applications such as automatic information extraction. As previous work on the SRL, there have been many machine learning approaches. (Gildea and Jurafsky, 2002; Pradhan et al., 2003; Lim et al., 2004).

In this paper, we present a two-phase SRL method based on a maximum entropy (ME) model. We first identify parse constituents that represent valid semantic arguments of a given predicate, and then assign appropriate semantic roles to the the identified parse constituents. In the two-phase SRL method, the performance of the argument identification phase is very important, because the argument classification is performed on the region identified at the identification phase. In this study, in order to improve the performance of identification, we try to incorporate *clause boundary* restriction and *tree distance* restriction into pre-processing of the identification phase.

Since features for identifying arguments are different from features for classifying a role, we need to determine different feature sets appropriate for the tasks. We determine final feature sets for each phase with experiments. We participate in the closed challenge of the CoNLL-2005 shared task and report results on both development and test sets. A detailed description of the task, data and related work can be found in Carreras and Màrquez (2005).

2 System Description

In this section, we describe our system that identifies and classifies semantic arguments. First, we explain pre-processing of the identification phase. Next, we describe features employed. Finally, we explain classifiers used in each phase.

2.1 Pre-processing

We thought that the occurrence of most semantic arguments are sensitive to the boundary of the immediate clause or the upper clauses of a predicate. Also, we assumed that they exist in the uniform distance on the parse tree from the predicate's parent node (called P_p) to the parse constituent's parent node (called P_c). Therefore, for identifying semantic arguments, we do not need to examine all parse constituents in a parse tree. In this study, we use the *clause boundary* restriction and the *tree distance* restriction, and they can provide useful information for spotting the probable search space which include semantic arguments.

In Figure 1 and Table 1, we show an example of applying the *tree distance* restriction. We show the distance between P_p=VP and the nonterminals of a parse tree in Figure 1. For example, NP_2:*d=3* means 3 times downward movement through the parse tree from P_p=VP to P_c=NP_2. NP_4 does not have the distance from P_p because we allow to move only upward or only downward through the tree from P_p to P_c. In Table 1, we indicate all 14 argument candidates that correspond to *tree distance* restriction (d$\leq$3). Only 2 of the 14 argument candidates are actually served to semantic arguments (NP_4, PP).

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 209–212, Ann Arbor, June 2005.

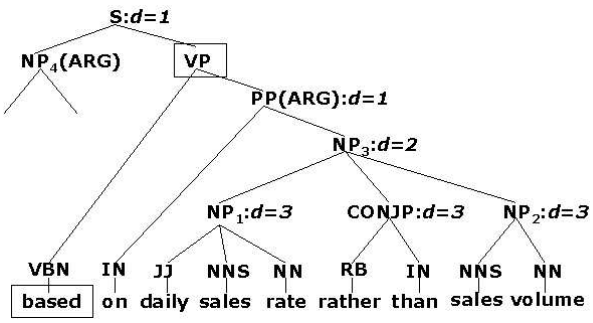


Figure 1: Distance between P_p=VP and P_c.

distance	direction	P_c	argument candidates
d=1	UP	S	NP_4
d=0	-	VP	PP
d=1	DOWN	PP	IN, NP_3
d=2	DOWN	NP_3	NP_1, CONJP, NP_2
d=3	DOWN	NP_1	JJ, NNS, NN
d=3	DOWN	CONJP	RB, IN
d=3	DOWN	NP_2	NNS, NN

Table 1: Probable argument candidates (d$\leq$3).

2.2 Features

The following features describe properties of the verb predicate. These featues are shared by all the parse constituents in the tree.

- *pred_lex*: this is the predicate itself.
- *pred_POS*: this is POS of the predicate.
- *pred_phr*: this is the syntactic category of P_p.
- *pred_type*: this represents the predicate usage such as to-infinitive form, the verb predicate of a main clause, and otherwise.
- *voice*: this is a binary feature identifying whether the predicate is active or passive.
- *sub_cat*: this is the phrase structure rule expanding the predicate's parent node in the tree.
- *pt+pl*: this is a conjoined feature of *pred_type* and *pred_lex*. Because the maximum entropy model assumes the independence of features, we need to conjoin the coherent features.

The following features characterize the internal structure of a argument candidate. These features change with the constituent under consideration.

- *head_lex*: this is the headword of the argument candidate. We extracts the headword by using the Collins's headword rules.
- *head_POS*: this is POS of the headword.
- *head_phr*: this is the syntactic category of P_c.
- *cont_lex*: this is the content word of the argument candidate. We extracts the content word by using the *head table* of the chunklink.pl [1].
- *cont_POS*: this is POS of the content word.
- *gov*: this is the *governing category* introduced by Gildea and Jurafsky (2002).

The following features capture the relations between the verb predicate and the constituent.

- *path*: this is the syntactic path through the parse tree from the parse constituent to the predicate.
- *pos*: this is a binary feature identifying whether the constituent is before or after the predicate.
- *pos+clau*: this, conjoined with *pos*, indicates whether the constituent is located in the immediate clause, in the first upper clause, in the second upper clause, or in the third upper clause.
- *pos+VP*, *pos+NP*, *pos+SBAR*: these are numeric features representing the number of the specific chunk types between the constituent and the predicate.
- *pos+CC*, *pos+comma*, *pos+colon*, *pos+quote*: these are numeric features representing the number of the specific POS types between the constituent and the predicate .
- *pl+hl* (*pred_lex* + *head_lex*), *pl+cl* (*pred_lex* + *cont_lex*), *v+gov* (*voice* + *gov*).

2.3 Classifier

The ME classifier for the identification phase classifies each parse constituent into one of the following classes: *ARG* class or *NON-ARG* class. The ME classifier for the classification phase classifies the identified argument into one of the pre-defined semantic roles (e.g. *A0*, *A1*, *AM-ADV*, *AM-CAU*, etc.).

[1] http://pi0657.kub.nl/šabine/chunklink/chunklink_2-2-2000_for_conll.pl

	#exa.	%can.	#can.	%arg.	$F_{\beta=1}$
no restriction					
All_1	3,709,080	-	233,394	96.06	79.37
All_2	2,579,278	-	233,004	95.90	79.52
All_3	1,598,726	100.00	231,120	95.13	79.92
restriction on clause boundary					
1/0	1,303,596	81.54	222,238	91.47	78.97
1/1	1,370,760	85.74	223,571	92.02	79.14
2/0	1,403,630	87.80	228,891	94.21	79.66
2/1	1,470,794	92.00	230,224	94.76	79.89
3/0	1,439,755	90.06	229,548	94.48	79.63
3/1	1,506,919	94.26	230,881	95.03	79.79
restriction on tree distance					
6/1	804,413	50.32	226,875	93.38	80.17
6/2	936,021	58.55	227,637	93.69	79.94
7/1	**842,453**	**52.70**	**228,129**	**93.90**	**80.44**
7/2	974,061	60.93	228,891	94.21	80.03
8/1	871,541	54.51	228,795	94.17	80.24
8/2	1,003,149	62.75	229,557	94.48	80.04
restriction on clause boundary & tree distance					
2/1,7/1	786,951	49.22	227,523	93.65	80.12
2/1,8/1	803,040	50.23	228,081	93.88	80.11
3/1,7/1	800,740	50.09	227,947	93.82	80.28
3/1,8/1	822,225	51.43	228,599	94.09	80.06

Table 2: Different ways of reducing candidates.

	Prec.	Recall	$F_{\beta=1}$	Accu.
All	82.57	78.41	80.44	86.00
All-(pred_lex)	82.80	77.78	80.21	84.93
All-(pred_POS)	83.40	76.72	79.92	85.95
All-(pred_phr)	83.11	77.57	80.24	85.87
All-(pred_type)	82.76	77.91	80.26	**85.99**
All-(voice)	82.87	77.88	80.30	85.88
All-(sub_cat)	82.48	77.68	80.00	84.88
All-(pt+pl)	83.20	77.40	80.20	85.62
All-(head_lex)	82.58	77.87	80.16	85.61
All-(head_POS)	82.66	77.88	80.20	85.89
All-(head_phr)	83.52	76.82	80.03	85.81
All-(cont_lex)	82.57	77.87	80.15	85.64
All-(cont_POS)	82.65	77.92	80.22	**86.09**
All-(gov)	82.69	78.34	**80.46**	85.91
All-(path)	78.39	67.96	72.80	85.69
All-(pos)	82.70	77.74	80.14	85.85
All-(pos+clau)	82.94	78.34	**80.57**	**86.19**
All-(pos+VP)	82.69	77.87	80.20	85.87
All-(pos+NP)	82.78	77.69	80.15	85.77
All-(pos+SBAR)	82.51	78.00	80.19	85.83
All-(pos+CC)	82.84	78.10	80.40	85.70
All-(pos+comma)	82.78	77.69	80.15	85.70
All-(pos+colon)	82.67	77.96	80.25	85.72
All-(pos+quote)	82.63	77.98	80.24	85.66
All-(pl+hl)	82.62	77.71	80.09	84.98
All-(pl+cl)	82.72	77.79	80.18	85.24
All-(v+gov)	82.93	77.81	80.29	85.85

	Prec.	Recall	$F_{\beta=1}$	Accu.
Iden.	82.56	78.72	80.59	-
clas.	-	-	-	87.16
Iden.+Clas.	72.68	69.16	70.87	-

Table 3: Performance of various feature combinations (top) and performance of each phase (bottom).

3 Experiments

To test the proposed method, we have experimented with CoNLL-2005 datasets (Wall Street sections 02-21 as training set, Charniak' trees). The results have been evaluated by using the *srl-eval.pl* script provided by the shared task organizers. For building classifiers, we utilized the Zhang le's MaxEnt toolkit [2], and the L-BFGS parameter estimation algorithm with Gaussian Prior smoothing.

Table 2 shows the different ways of reducing the number of argument candidates. The 2nd and 3rd columns (*#can.*, *%can.*) indicate the number of argument candidates and the percentage of argument candidates that satisfy each restriction on the training set. The 4th and 5th columns (*#arg.*, *%arg.*) indicate the number of correct arguments and the percentage of correct arguments that satisfy each restriction on the training set. The last column ($F_{\beta=1}$) indicates the performance of the identification task on the development set by applying each restriction.

In *no restriction*, All_1 extracts candidates from all the nonterminals's child nodes of a tree. All_2 filter the nonterminals which include at least one nonterminal child [3]. All_3 filter the nonterminals which include at least one nonterminal child and have distance from P_p. We use All_3 as a baseline.

In *restriction on clause boundary*, for example, *2/0* means that the left search boundary for identifying the argument is set to the left boundary of the second upper clause, and the right search boundary is set to the right boundary of the immediate clause.

In *restriction on tree distance*, for example, *7/1* means that it is possible to move up to 7 times upward ($d \leq 7$) through the parse tree from P_p to P_c, and it is possible to move up to once downward ($d \leq 1$) through the parse tree from P_p to P_c.

In *clause boundary & tree distance*, for example, *3/1,7/1* means the case when we use both the *clause boundary* (*3/1*) and the *tree distance* (*7/1*).

[2] http://www.nlplab.cn/zhangle/maxent_toolkit.html

[3] We ignore the nonterminals that have only pre-terminal children (e.g. in Figure 1, NP_1, CONJP, NP_2).

	Precision	Recall	$F_{\beta=1}$
Development	72.68%	69.16%	70.87
Test WSJ	74.69%	70.78%	72.68
Test Brown	64.58%	60.31%	62.38
Test WSJ+Brown	73.35%	69.37%	71.31

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	74.69%	70.78%	72.68
A0	85.02%	81.53%	83.24
A1	73.98%	72.25%	73.11
A2	63.20%	57.57%	60.25
A3	62.96%	49.13%	55.19
A4	73.40%	67.65%	70.41
A5	100.00%	40.00%	57.14
AM-ADV	56.73%	50.00%	53.15
AM-CAU	70.21%	45.21%	55.00
AM-DIR	46.48%	38.82%	42.31
AM-DIS	70.95%	65.62%	68.18
AM-EXT	87.50%	43.75%	58.33
AM-LOC	44.09%	46.28%	45.16
AM-MNR	55.56%	52.33%	53.89
AM-MOD	97.59%	95.64%	96.61
AM-NEG	96.05%	95.22%	95.63
AM-PNC	40.68%	41.74%	41.20
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	70.11%	61.73%	65.66
R-A0	84.68%	83.93%	84.30
R-A1	73.33%	70.51%	71.90
R-A2	50.00%	31.25%	38.46
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	57.14%	68.57
R-AM-MNR	16.67%	16.67%	16.67
R-AM-TMP	72.50%	55.77%	63.04
V	97.32%	97.32%	97.32

Table 4: Overall results (top) and detailed results on the WSJ test (bottom).

	Precision	Recall	$F_{\beta=1}$
one-phase	71.94	68.70	70.29
two-phase	72.68	69.16	70.87

Table 5: Performance of one-phase vs. two-phase.

According to the experimental results, we use *7/1 tree distance* restriction for all following experiments. By applying the restriction, we can remove about 47.3% (%can.=52.70%) of total argument candidates as compared with All_3. 93.90% (%arg.) corresponds to the upper bound on recall.

In order to estimate the relative contribution of each feature, we measure performance of each phase on the development set by leaving out one feature at a time, as shown in the top of Table 3. *Precision*, *Recall*, and $F_{\beta=1}$ represent the performance of the identification task, and *Accuracy* represent the performance of the classification task only with 100% correct argument identification respectively. *All* represents the performance of the experiment when all 26 features introduced by section 2.2 are considered. Finally, for identification, we use 24 features except *gov* and *pos+clau*, and obtain an $F_{\beta=1}$ of 80.59%, as shown in the bottom of Table 3. Also, for classification, we use 23 features except *pred_type*, *cont_POS*, and *pos+clau*, and obtain an *Accuracy* of 87.16%.

Table 4 presents our best system performance on the development set, and the performance of the same system on the test set. Table 5 shows the performance on the development set using the one-phase method and the two-phase method respectively. The one-phase method is implemented by incorporating the identification into the classification. *one-phase* shows the performance of the experiment when 25 features except *pos+clau* are used. Experimental results show that the two-phase method is better than the one-phase method in our evaluation.

4 Conclusion

We have presented a two-phase SRL method based on a ME model. In the two-phase method, in order to improve the performance of identification that dominate the overall performance, we have performed pre-processing. Experimental results show that our system obtains an $F_{\beta=1}$ of 72.68% on the WSJ test and that the introduction of pre-processing improves the performance, as compared with the case when all parse constituents are considered.

References

Xavier Carreras and Lluís Màrquez. 2005. *Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling*. Proceedings of CoNLL-2005.

Daniel Gildea and Daniel Jurafsky. 2002. *Automatic Labeling of Semantic Roles*. Computational Linguistics.

Joon-Ho Lim, Young-Sook Hwang, So-Young Park and Hae-Chang Rim. 2004. *Semantic Role Labeling using Maximum Entropy Model*. Proceedings of CoNLL.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin and Daniel Jurafsky. 2003. *Shallow Semantic Parsing Using Support Vector Machines*. Technical Report, TR-CSLR-2003-03.

Semantic Role Labeling Using Lexical Statistical Information

Simone Paolo Ponzetto and **Michael Strube**
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
http://www.eml-research.de/nlp/

Abstract

Our system for semantic role labeling is multi-stage in nature, being based on tree pruning techniques, statistical methods for lexicalised feature encoding, and a C4.5 decision tree classifier. We use both shallow and deep syntactic information from automatically generated chunks and parse trees, and develop a model for learning the semantic arguments of predicates as a multi-class decision problem. We evaluate the performance on a set of relatively 'cheap' features and report an F_1 score of 68.13% on the overall test set.

1 Introduction

This paper presents a system for the CoNLL 2005 Semantic Role Labeling shared task (Carreras & Màrquez, 2005), which is based on the current release of the English PropBank data (Palmer et al., 2005). For the 2005 edition of the shared task are available both syntactic and semantic information. Accordingly, we make use of both clausal, chunk and deep syntactic (tree structure) features, named entity information, as well as statistical representations for lexical item encoding.

The set of features and their encoding reflect the necessity of limiting the complexity and dimensionality of the input space. They also provide the classifier with enough information. We explore here the use of a minimal set of compact features for semantic role prediction, and show that a feature-based statistical encoding of lexicalised features such as predicates, head words, local contexts and PoS by means of probability distributions provides an efficient way of representing the data, with the feature vectors having a small dimensionality and allowing to abstract from single words.

2 System description

2.1 Preprocessing

During preprocessing the predicates' semantic arguments are mapped to the nodes in the parse trees, a set of hand-crafted shallow tree pruning rules are applied, probability distributions for feature representation are generated from training data[1], and feature vectors are extracted. Those are finally fed into the classifier for semantic role classification.

2.1.1 Tree node mapping of semantic arguments and named entities

Following Gildea & Jurafsky (2002), (i) labels matching more than one constituent due to non-branching nodes are taken as labels of higher constituents, (ii) in cases of labels with no corresponding parse constituent, these are assigned to the partial match given by the constituent spanning the shortest portion of the sentence beginning at the label's span left boundary and lying entirely within it. We drop the role or named entity label if such suitable constituent could not be found[2].

[1] All other processing steps assume a uniform treatment of both training and test data.

[2] The percentage of roles for which no valid tree node could be found amounts to 3% for the training and 7% for the development set. These results are compatible with the performance of the employed parser (Collins, 1999).

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 213–216, Ann Arbor, June 2005.

2.1.2 Tree pruning

The tagged trees are further processed by applying the following pruning rules:

- All punctuation nodes are removed. This is for removing punctuation information, as well as for aligning spans of the syntactic nodes with PropBank constituents[3].
- If a node is unary branching and its daughter is also unary branching, the daughter is removed. This allows to remove redundant nodes spanning the same tokens in the sentence.
- If a node has only preterminal children, these are removed. This allows to internally collapse base phrases such as base NPs.

Tree pruning was carried out in order to reduce the number of nodes from which features were to be extracted later. This limits the number of candidate constituents for role labeling, and removes redundant information produced by the pipeline of previous components (i.e. PoS tags of preterminal labels), as well as the sparseness and fragmentation of the input data. These simple rules reduce the number of constituents given by the parser output by 38.4% on the training set, and by 38.7% on the development set, at the cost of limiting the coverage of the system by removing approximately 2% of the target role labeled constituents. On the development set, the number of constituents remaining on top of pruning is 81,193 of which 7,558 are semantic arguments, with a performance upper-bound of 90.6% F_1.

2.1.3 Features

Given the pruned tree structures, we traverse the tree bottom-up left-to-right. For each non-terminal node whose span does not overlap the predicate we extract the following features:

Phrase type: the syntactic category of the constituent (NP, PP, ADVP, etc.). In order to reduce the number of phrase labels, we retained only those labels which account for at least 0.1% of the overall available semantic arguments in the training data. We replace the label for every phrase type category below this threshold with a generic `UNK` label. This reduces the number of labels from 72 to 18.

Position: the position of the constituent with respect to the target predicate (*before* or *after*).

Adjacency: whether the right (if before) or left (if after) boundary of the constituent is *adjacent*, *non-adjacent* or *inside* the predicate's chunk.

Clause: whether the constituent belongs to the clause of the predicate or not.

Proposition size: measures relative to the proposition size, such as (i) the number of constituents and (ii) predicates in the proposition.

Constituent size: measures relative to the constituent size, namely (i) the number of tokens and (ii) subconstituents (viz., non-leaf rooted subtrees) of the constituent.

Predicate: the predicate lemma, represented as the probability distribution $P(r|p)$ of the predicate p of taking one of the available r semantic roles. For unseen predicates we assume a uniform distribution.

Voice: whether the predicate is in *active* or *passive* form. Passive voice is identified if the predicate's PoS tag is `VBN` and either it follows a form of *to be* or *to get*, or it does not belong to a VP chunk, or is immediately preceded by an NP chunk.

Head word: the head word of the constituent, represented as the probability distribution $P(r|hw)$ of the head word hw of heading a phrase filling one of the available r semantic roles. For unseen words we back off on a phrasal model by using the probability distribution $P(r|pt)$ of the phrase type pt of filling a semantic slot r.

Head word PoS: the PoS of the head word of the constituent, similarly represented as the probability distribution $P(r|pos)$ of a PoS pos of belonging to a constituent filling one of the available r semantic roles.

Local lexical context: the words in the constituent other than the head word, represented as the

[3] We noted during prototyping that in many cases no tree node fully matching a role constituent could be found, as the latter did not include punctuation tokens, whereas in Collins' trees the punctuation terminals are included within the preceding phrases. This precludes *a priori* the output to align to the gold standard PropBank annotation and we use therefore pruning as a recovery strategy.

averaged probability distributions of each i-th non-head word w_i of occurring in one of the available r semantic roles, namely $\frac{1}{m}\sum_{i=1}^{m} P(r|w_i)$ for m non-head words in the constituent. For each unseen word we back off by using the probability distribution $P(r|pos_i)$ of the PoS pos_i of filling a semantic role r[4].

Named entities: the label of the named entity which spans the same words as the constituent, as well as the label of the largest named entity *embedded* within the constituent. Both values are set to `NULL` if such labels could not be found.

Path: the number of intervening NPB, NP, VP, VP-A, PP, PP-A, S, S-A and SBAR nodes along the path from the constituent to the predicate.

Distance: the distance from the target predicate, measured as (i) the number of nodes from the constituent to the lowest node in the tree dominating both the constituent and the predicate, (ii) the number of nodes from the predicate to the former common dominating node[5], (iii) the number of chunks between the base phrase of the constituent's head and the predicate chunk, (iv) the number of tokens between the head of the constituent and the predicate.

2.2 Classifier

We used the YaDT[6] implementation of the C4.5 decision tree algorithm (Quinlan, 1993). Parameter selection (99% pruning confidence, at least 10 instances per leaf node) was carried out by performing 10-fold cross-validation on the development set.

Data preprocessing and feature vector generation took approximately 2.5 hours (training set, including probability distribution generation), 5 minutes (development) and 7 minutes (test) on a 2GHz Opteron dual processor server with 2GB memory[7]. Training time was of approximately 17 minutes. The final system was trained using all of the available training data from sections 2–21 of the Penn TreeBank. This amounts to 2,250,887 input constituents of which 10% are non-`NULL` examples. Interestingly, during prototyping we first limited ourselves to training and drawing probability distributions for feature representation from sections 15–18 only. This yielded a very low performance (57.23% F_1, development set). A substantial performance increase was given by still training on sections 15–18, but using the probability distributions generated from sections 2–21 (64.43% F_1, development set). This suggests that the system is only marginally sensitive to the training dataset size, but pivotally relies on taking probability distributions from a large amount of data.

In order to make the task easier and overcome the uneven role class distribution, we limited the learner to classify only those 16 roles accounting for at least 0.5% of the total number of semantic arguments in the training data[8].

2.3 Post-processing

As our system does not build an overall sentence contextual representation, it systematically produced errors such as embedded role labeling. In particular, since no embedding is observed for the semantic arguments of predicates, in case of (multiple) embeddings the classifier output was automatically post-processed to retain only the largest embedding constituent. Evaluation on the development set has shown that this does not significantly improve performance, still it provides a much more 'sane' output. Besides, we make use of a simple technique for avoiding multiple `A0` or `A1` role assignments within the same proposition, based on constituent position and predicate voice. In case of multiple `A0` labels, if the predicate is in active form, the second `A0` occurrence is replaced with `A1`, else we replace the first occurrence. Similarly, in case of multiple `A1` labels, if the predicate is in active form, the first `A1` occurrence is replaced with `A0`, else we

[4]This feature was introduced as the information provided by lexical heads does not seem to suffice in many cases. This is shown by head word ambiguities, such as `LOC` and `TMP` arguments occurring in similar prepositional syntactic configurations — i.e. the preposition *in*, which can be head of both `AM-TMP` and `AM-LOC` constituents, as in *in October* and *in New York*. The idea is therefore to look at the words in the constituents other than the head, and build up an overall constituent representation, thus making use of statistical lexical information for role disambiguation.

[5]These distance measures along the tree path between the constituent and the predicate were kept separate, in order to indirectly include *embedding level* information into the model.

[6]`http://www.di.unipi.it/~ruggieri/software.html`

[7]We used only a single CPU at runtime, since the implementation is not parallelised.

[8]These include numbered arguments (`A0` to `A4`), adjuncts (`ADV`, `DIS`, `LOC`, `MNR`, `MOD`, `NEG`, `PNC`, `TMP`), and references (`R-A0` and `R-A1`).

	Precision	Recall	$F_{\beta=1}$
Development	71.82%	61.60%	66.32
Test WSJ	75.05%	64.81%	69.56
Test Brown	66.69%	52.14%	58.52
Test WSJ+Brown	74.02%	63.12%	68.13

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	75.05%	64.81%	69.56
A0	78.52%	72.52%	75.40
A1	75.53%	65.39%	70.10
A2	62.28%	52.07%	56.72
A3	63.81%	38.73%	48.20
A4	73.03%	63.73%	68.06
A5	0.00%	0.00%	0.00
AM-ADV	60.00%	42.69%	49.88
AM-CAU	0.00%	0.00%	0.00
AM-DIR	0.00%	0.00%	0.00
AM-DIS	75.97%	73.12%	74.52
AM-EXT	0.00%	0.00%	0.00
AM-LOC	54.09%	47.38%	50.51
AM-MNR	58.67%	46.22%	51.71
AM-MOD	97.43%	96.37%	96.90
AM-NEG	97.78%	95.65%	96.70
AM-PNC	42.17%	30.43%	35.35
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	75.41%	71.11%	73.20
R-A0	82.09%	73.66%	77.65
R-A1	72.03%	66.03%	68.90
R-A2	0.00%	0.00%	0.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	0.00%	0.00%	0.00
V	98.63%	98.63%	98.63

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

replace the second occurrence.

3 Results

Table 1 shows the results on the test set. Problems are inherently related with the skewed distribution of role classes, so that roles which have a limited number of occurrences are harder to classify correctly. This explains the performance gap on the A0 and A1 roles on one hand, and the A2, A3, A4, AM- arguments on the other.

One advantage of using a decision tree learning algorithm is that it outputs a model which includes a feature ranking, since the most informative features are those close to the root of the tree. In the present case, the most informative features were both distance/position metrics (distance and adjacency) and lexicalized features (head word and predicate).

4 Conclusion

Semantic role labeling is a difficult task, and accordingly, how to achieve an accurate and robust performance is still an open question. In our work we used a limited set of syntactic tree based distance and size metrics coupled with raw lexical statistics, and showed that such 'lazy learning' configuration can still achieve a reasonable performance.

We concentrated on reducing the complexity given by the number and dimensionality of the instances to be classified during learning. This is the core motivation behind performing tree pruning and statistical feature encoding. This also helped us to avoid the use of sparse features such as the explicit path in the parse tree between the candidate constituent and the predicate, and the predicate's subcategorization rule (cf. e.g. Pradhan et al. (2004)).

Future work will concentrate on benchmarking this approach within alternative architectures (i.e. two-phase with filtering) and different learning schemes (i.e. vector-based methods such as Support Vector Machines and Artificial Neural Networks).

Acknowledgements: This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a KTF grant (09.003.2004).

References

Carreras, Xavier & Lluís Màrquez (2005). Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Collins, Michael (1999). *Head-driven statistical models for natural language parsing*, (Ph.D. thesis). Philadelphia, Penn., USA: University of Pennsylvania.

Gildea, Daniel & Daniel Jurafsky (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Palmer, Martha, Dan Gildea & Paul Kingsbury (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Pradhan, Sameer, Kadri Hacioglu, Valeri Krugler, Wayne Ward, James H. Martin & Daniel Jurafsky (2004). Support vector learning for semantic argument classification. *Journal of Machine Learning, Special issue on Speech and Natural Language Processing*. To appear.

Quinlan, J. Ross (1993). *C4.5: programs for machine learning*. San Francisco, Cal., USA: Morgan Kaufmann Publishers Inc.

Semantic Role Chunking Combining Complementary Syntactic Views

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin and Daniel Jurafsky[†]
Center for Spoken Language Research, University of Colorado, Boulder, CO 80303
[†]Department of Linguistics, Stanford University, Stanford, CA 94305
{spradhan,hacioglu,whw,martin}@cslr.colorado.edu, jurafsky@stanford.edu

Abstract

This paper describes a semantic role labeling system that uses features derived from different syntactic views, and combines them within a phrase-based chunking paradigm. For an input sentence, syntactic constituent structure parses are generated by a Charniak parser and a Collins parser. Semantic role labels are assigned to the constituents of each parse using Support Vector Machine classifiers. The resulting semantic role labels are converted to an IOB representation. These IOB representations are used as additional features, along with flat syntactic chunks, by a chunking SVM classifier that produces the final SRL output. This strategy for combining features from three different syntactic views gives a significant improvement in performance over roles produced by using any one of the syntactic views individually.

1 Introduction

The task of Semantic Role Labeling (SRL) involves tagging groups of words in a sentence with the semantic roles that they play with respect to a particular predicate in that sentence. Our approach is to use supervised machine learning classifiers to produce the role labels based on features extracted from the input. This approach is neutral to the particular set of labels used, and will learn to tag input according to the annotated data that it is trained on. The task reported on here is to produce PropBank (Kingsbury and Palmer, 2002) labels, given the features provided for the CoNLL-2005 closed task (Carreras and Màrquez, 2005).

We have previously reported on using SVM classifiers for semantic role labeling. In this work, we formulate the semantic labeling problem as a multi-class classification problem using Support Vector Machine (SVM) classifiers. Some of these systems use features based on syntactic constituents produced by a Charniak parser (Pradhan et al., 2003; Pradhan et al., 2004) and others use only a flat syntactic representation produced by a syntactic chunker (Hacioglu et al., 2003; Hacioglu and Ward, 2003; Hacioglu, 2004; Hacioglu et al., 2004). The latter approach lacks the information provided by the hierarchical syntactic structure, and the former imposes a limitation that the possible candidate roles should be one of the nodes already present in the syntax tree. We found that, while the chunk based systems are very efficient and robust, the systems that use features based on full syntactic parses are generally more accurate. Analysis of the source of errors for the parse constituent based systems showed that incorrect parses were a major source of error. The syntactic parser did not produce any constituent that corresponded to the correct segmentation for the semantic argument. In Pradhan et al. (2005), we reported on a first attempt to overcome this problem by combining semantic role labels produced from different syntactic parses. The hope is that the syntactic parsers will make different errors, and that combining their outputs will improve on

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 217–220, Ann Arbor, June 2005.

either system alone. This initial attempt used features from a Charniak parser, a Minipar parser and a chunk based parser. It did show some improvement from the combination, but the method for combining the information was heuristic and sub-optimal. In this paper, we report on what we believe is an improved framework for combining information from different syntactic views. Our goal is to preserve the robustness and flexibility of the segmentation of the phrase-based chunker, but to take advantage of features from full syntactic parses. We also want to combine features from different syntactic parses to gain additional robustness. To this end, we use features generated from a Charniak parser and a Collins parser, as supplied for the CoNLL-2005 closed task.

2 System Description

We again formulate the semantic labeling problem as a multi-class classification problem using Support Vector Machine (SVM) classifiers. TinySVM[1] along with YamCha[2] (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2001) are used to implement the system. Using what is known as the ONE VS ALL classification strategy, n binary classifiers are trained, where n is number of semantic classes including a NULL class.

The general framework is to train separate semantic role labeling systems for each of the parse tree views, and then to use the role arguments output by these systems as additional features in a semantic role classifier using a flat syntactic view. The constituent based classifiers walk a syntactic parse tree and classify each node as NULL (no role) or as one of the set of semantic roles. Chunk based systems classify each base phrase as being the B(eginning) of a semantic role, I(nside) a semantic role, or O(utside) any semantic role (ie. NULL). This is referred to as an IOB representation (Ramshaw and Marcus, 1995). The constituent level roles are mapped to the IOB representation used by the chunker. The IOB tags are then used as features for a separate base-phase semantic role labeler (chunker), in addition to the standard set of features used by the chunker. An n-fold cross-validation paradigm is used to train the constituent based role classifiers and the chunk based classifier.

For the system reported here, two full syntactic parsers were used, a Charniak parser and a Collins parser. Features were extracted by first generating the Collins and Charniak syntax trees from the word-by-word decomposed trees in the CoNLL data. The chunking system for combining all features was trained using a 4-fold paradigm. In each fold, separate SVM classifiers were trained for the Collins and Charniak parses using 75% of the training data. That is, one system assigned role labels to the nodes in Charniak based trees and a separate system assigned roles to nodes in Collins based trees. The other 25% of the training data was then labeled by each of the systems. Iterating this process 4 times created the training set for the chunker. After the chunker was trained, the Charniak and Collins based semantic labelers were then retrained using all of the training data.

Two pieces of the system have problems scaling to large training sets – the final chunk based classifier and the NULL VS NON-NULL classifier for the parse tree syntactic views. Two techniques were used to reduce the amount of training data – active sampling and NULL filtering. The active sampling process was performed as follows. We first train a system using 10k seed examples from the training set. We then labeled an additional block of data using this system. Any sentences containing an error were added to the seed training set. The system was retrained and the procedure repeated until there were no misclassified sentences remaining in the training data. The set of examples produced by this procedure was used to train the final NULL VS NON-NULL classifier. The same procedure was carried out for the chunking system. After both these were trained, we tagged the training data using them and removed all most likely NULLs from the data.

Table 1 lists the features used in the constituent based systems. They are a combination of features introduced by Gildea and Jurafsky (2002), ones proposed in Pradhan et al. (2004), Surdeanu et al. (2003) and the *syntactic-frame* feature proposed in (Xue and Palmer, 2004). These features are extracted from the parse tree being labeled. In addition to the features extracted from the parse tree being labeled, five features were extracted from the other parse tree (phrase, head word, head word POS, path

[1] http://chasen.org/~taku/software/TinySVM/

[2] http://chasen.org/~taku/software/yamcha/

PREDICATE LEMMA
PATH: Path from the constituent to the predicate in the parse tree.
POSITION: Whether the constituent is before or after the predicate.
PREDICATE SUB-CATEGORIZATION
HEAD WORD: Head word of the constituent.
HEAD WORD POS: POS of the head word
NAMED ENTITIES IN CONSTITUENTS: Person, Organization, Location and Miscellaneous.
PARTIAL PATH: Path from the constituent to the lowest common ancestor of the predicate and the constituent.
HEAD WORD OF PP: Head of PP replaced by head word of NP inside it, and PP replaced by PP-*preposition*
FIRST AND LAST WORD/POS IN CONSTITUENT
ORDINAL CONSTITUENT POSITION
CONSTITUENT TREE DISTANCE
CONSTITUENT RELATIVE FEATURES: Nine features representing the phrase type, head word and head word part of speech of the parent, and left and right siblings of the constituent.
SYNTACTIC FRAME
CONTENT WORD FEATURES: Content word, its POS and named entities in the content word
CLAUSE-BASED PATH VARIATIONS: I. Replacing all the nodes in a path other than clause nodes with an "*". For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes NP↑S↑*S↑*↑*↓VBD II. Retaining only the clause nodes in the path, which for the above example would produce NP↑S↑S↓VBD, III. Adding a binary feature that indicates whether the constituent is in the same clause as the predicate, IV. collapsing the nodes between S nodes which gives NP↑S↑NP↑VP↓VBD.
PATH N-GRAMS: This feature decomposes a path into a series of trigrams. For example, the path NP↑S↑VP↑SBAR↑NP↑VP↓VBD becomes: NP↑S↑VP, S↑VP↑SBAR, VP↑SBAR↑NP, SBAR↑NP↑VP, etc. We used the first ten trigrams as ten features. Shorter paths were padded with nulls.
SINGLE CHARACTER PHRASE TAGS: Each phrase category is clustered to a category defined by the first character of the phrase label.
PREDICATE CONTEXT: Two words and two word POS around the predicate and including the predicate were added as ten new features.
PUNCTUATION: Punctuation before and after the constituent were added as two new features.
FEATURE CONTEXT: Features for argument bearing constituents were added as features to the constituent being classified.

Table 1: Features used by the constituent-based system

and predicate sub-categorization). So for example, when assigning labels to constituents in a Charniak parse, all of the features in Table 1 were extracted from the Charniak tree, and in addition phrase, head word, head word POS, path and sub-categorization were extracted from the Collins tree. We have previously determined that using different sets of features for each argument (role) achieves better results than using the same set of features for all argument classes. A simple feature selection was implemented by adding features one by one to an initial set of features and selecting those that contribute significantly to the performance. As described in Pradhan et al. (2004), we post-process lattices of n-best decision using a trigram language model of argument sequences.

Table 2 lists the features used by the chunker. These are the same set of features that were used in the CoNLL-2004 semantic role labeling task by Hacioglu, et al. (2004) with the addition of the two semantic argument (IOB) features. For each token (base phrase) to be tagged, a set of features is created from a fixed size context that surrounds each token. In addition to the features in Table 2, it also uses previous semantic tags that have already been assigned to the tokens contained in the linguistic context. A 5-token sliding window is used for the context.

SVMs were trained for begin (B) and inside (I) classes of all arguments and an outside (O) class.

WORDS
PREDICATE LEMMAS
PART OF SPEECH TAGS
BP POSITIONS: The position of a token in a BP using the IOB2 representation (e.g. B-NP, I-NP, O, etc.)
CLAUSE TAGS: The tags that mark token positions in a sentence with respect to clauses.
NAMED ENTITIES: The IOB tags of named entities.
TOKEN POSITION: The position of the phrase with respect to the predicate. It has three values as "before", "after" and "-" (for the predicate)
PATH: It defines a flat path between the token and the predicate
HIERARCHICAL PATH: Since we have the syntax tree for the sentences, we also use the hierarchical path from the phrase being classified to the base phrase containing the predicate.
CLAUSE BRACKET PATTERNS
CLAUSE POSITION: A binary feature that identifies whether the token is inside or outside the clause containing the predicate
HEADWORD SUFFIXES: suffixes of headwords of length 2, 3 and 4.
DISTANCE: Distance of the token from the predicate as a number of base phrases, and the distance as the number of VP chunks.
LENGTH: the number of words in a token.
PREDICATE POS TAG: the part of speech category of the predicate
PREDICATE FREQUENCY: Frequent or rare using a threshold of 3.
PREDICATE BP CONTEXT: The chain of BPs centered at the predicate within a window of size -2/+2.
PREDICATE POS CONTEXT: POS tags of words immediately preceding and following the predicate.
PREDICATE ARGUMENT FRAMES: Left and right core argument patterns around the predicate.
DYNAMIC CLASS CONTEXT: Hypotheses generated for two preceeding phrases.
NUMBER OF PREDICATES: This is the number of predicates in the sentence.
CHARNIAK-BASED SEMANTIC IOB TAG: This is the IOB tag generated using the tagger trained on Charniak trees
COLLINS-BASED SEMANTIC IOB TAG: This is the IOB tag generated using the tagger trained on Collins' trees

Table 2: Features used by phrase-based chunker.

3 Experimental Results

Table 3 shows the results obtained on the WSJ development set (Section 24), the WSJ test set (Section 23) and the Brown test set (Section ck/01-03)

4 Acknowledgments

This research was partially supported by the ARDA AQUAINT program via contract OCG4423B and by the NSF via grants IS-9978025 and ITR/HCI

	Precision	Recall	$F_{\beta=1}$
Development	80.90%	75.38%	78.04
Test WSJ	81.97%	73.27%	77.37
Test Brown	73.73%	61.51%	67.07
Test WSJ+Brown	80.93%	71.69%	76.03

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	81.97%	73.27%	77.37
A0	91.39%	82.23%	86.57
A1	79.80%	76.23%	77.97
A2	68.61%	62.61%	65.47
A3	73.95%	50.87%	60.27
A4	78.65%	68.63%	73.30
A5	75.00%	60.00%	66.67
AM-ADV	61.64%	46.05%	52.71
AM-CAU	76.19%	43.84%	55.65
AM-DIR	53.33%	37.65%	44.14
AM-DIS	80.56%	63.44%	70.98
AM-EXT	100.00%	46.88%	63.83
AM-LOC	64.48%	51.52%	57.27
AM-MNR	62.90%	45.35%	52.70
AM-MOD	98.64%	92.38%	95.41
AM-NEG	98.21%	95.65%	96.92
AM-PNC	56.67%	44.35%	49.76
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	83.37%	71.94%	77.23
R-A0	94.29%	88.39%	91.24
R-A1	85.93%	74.36%	79.73
R-A2	100.00%	37.50%	54.55
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	90.00%	42.86%	58.06
R-AM-MNR	66.67%	33.33%	44.44
R-AM-TMP	75.00%	40.38%	52.50
V	98.86%	98.86%	98.86

Table 3: Overall results (top) and detailed results on the WSJ test (bottom).

0086132. Computer time was provided by NSF ARI Grant #CDA-9601817, NSF MRI Grant #CNS-0420873, NASA AIST grant #NAG2-1646, DOE SciDAC grant #DE-FG02-04ER63870, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research (SUR) program.

Special thanks to Matthew Woitaszek, Theron Voran and the other administrative team of the Hemisphere and Occam Beowulf clusters. Without these the training would never be possible.

References

Xavier Carreras and Lluís Màrquez. 2005. n Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Kadri Hacioglu and Wayne Ward. 2003. Target word detection and semantic role chunking using support vector machines. In *Proceedings of the Human Language Technology Conference*, Edmonton, Canada.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Shallow semantic parsing using support vector machines. Technical Report TR-CSLR-2003-1, Center for Spoken Language Research, Boulder, Colorado.

Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James Martin, and Daniel Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of the 8th Conference on CoNLL-2004, Shared Task – Semantic Role Labeling*.

Kadri Hacioglu. 2004. A lightweight semantic chunking model based on tagging. In *Proceedings of the Human Language Technology Conference /North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Canary Islands, Spain.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of the 4th Conference on CoNLL-2000 and LLL-2000*, pages 142–144.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James Martin, and Dan Jurafsky. 2003. Semantic role parsing: Adding semantic structure to unstructured text. In *Proceedings of the International Conference on Data Mining (ICDM 2003)*, Melbourne, Florida.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the Association for Computational Linguistics 43rd annual meeting (ACL-2005)*, Ann Arbor, MI.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Annual Workshop on Very Large Corpora*, pages 82–94. ACL.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain.

Semantic Role Labeling Using Complete Syntactic Analysis

Mihai Surdeanu
Technical University of Catalunya
surdeanu@lsi.upc.edu

Jordi Turmo
Technical University of Catalunya
turmo@lsi.upc.edu

Abstract

In this paper we introduce a semantic role labeling system constructed on top of the full syntactic analysis of text. The labeling problem is modeled using a rich set of lexical, syntactic, and semantic attributes and learned using one-versus-all AdaBoost classifiers.

Our results indicate that even a simple approach that assumes that each semantic argument maps into exactly one syntactic phrase obtains encouraging performance, surpassing the best system that uses partial syntax by almost 6%.

1 Introduction

Most current semantic role labeling (SRL) approaches can be classified in one of two classes: approaches that take advantage of complete syntactic analysis of text, pioneered by (Gildea and Jurafsky, 2002), and approaches that use partial syntactic analysis, championed by the previous CoNLL shared task evaluations (Carreras and Màrquez, 2004).

However, to the authors' knowledge, a clear analysis of the benefits of using full syntactic analysis versus partial analysis is not yet available. On one hand, the additional information provided by complete syntax should intuitively be useful. But, on the other hand, the state-of-the-art of full parsing is known to be less robust and perform worse than the tools used for partial syntactic analysis, which would decrease the quality of the information provided. The work presented in this paper contributes to this analysis by introducing a model that is entirely based on the full syntactic analysis of text, generated by a real-world parser.

2 System Description

2.1 Mapping Arguments to Syntactic Constituents

Our approach maps each argument label to one syntactic constituent, using a strategy similar to (Surdeanu et al., 2003). Using a bottom-up approach, we map each argument to the first phrase that has the exact same boundaries and climb as high as possible in the syntactic tree across unary production chains.

Unfortunately, this one-to-one mapping between semantic arguments and syntactic constituents is not always possible. One semantic argument may be mapped to many syntactic constituents due to: (a) intrinsic differences between the syntactic and semantic representations, and (b) incorrect syntactic structure. Figure 1 illustrates each one of these situations: Figure 1 (a) shows a sentence where each semantic argument correctly maps to one syntactic constituent; Figure 1 (b) illustrates the situation where one semantic argument correctly maps to two syntactic constituents; and Figure 1 (c) shows a one-to-many mapping caused by an incorrect syntactic structure: argument `A0` maps to two phrases, the terminal "by" and the noun phrase "Robert Goldberg", due to the incorrect attachment of the last prepositional phrase, "at the University of California".

Using the above observations, we separate one-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 221–224, Ann Arbor, June 2005.

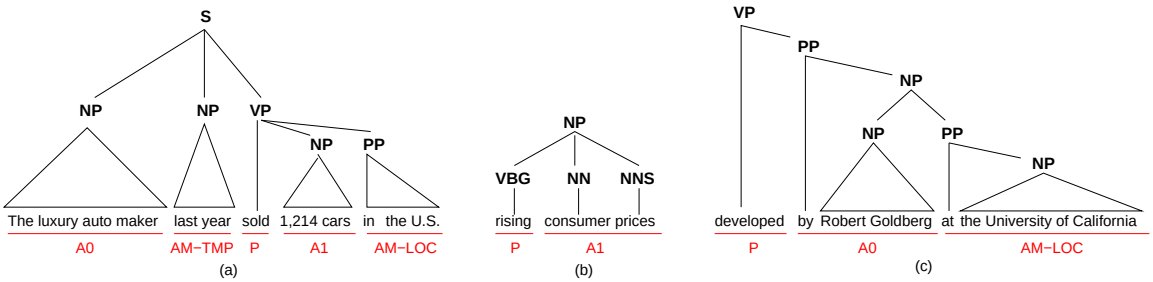


Figure 1: Mapping semantic arguments to syntactic constituents: (a) correct one–to-one mapping; (b) correct one-to-many mapping; (c) one-to-many mapping due to incorrect syntax.

	(a)	(b)	(c)
Training	96.06%	2.49%	1.45%
Development	91.36%	4.83%	3.81%

Table 1: Distribution of semantic arguments according to their mapping to syntactic constituents obtained with the Charniak parser: (a) one-to-one, (b) one-to-many, all syntactic constituents have same parent, (c) one-to-many, syntactic constituents have different parents.

to-many mappings in two classes: (a) when the syntactic constituents mapped to the semantic argument have the same parent (Figure 1 (b)) the mapping is correct and/or could theoretically be learned by a sequential SRL strategy, and (b) when the syntactic constituents mapped to the same argument have different parents, the mapping is generally caused by incorrect syntax. Such cases are very hard to be learned due to the irregularities of the parser errors.

Table 1 shows the distribution of semantic arguments into one of the above classes, using the syntactic trees provided by the Charniak parser. For the results reported in this paper, we model only one-to-one mappings between semantic arguments and syntactic constituents. A subset of the one-to-many mappings are addressed with a simple heuristic, described in Section 2.4.

2.2 Features

The features incorporated in the proposed model are inspired from the work of (Gildea and Jurafsky, 2002; Surdeanu et al., 2003; Pradhan et al., 2005; Collins, 1999) and can be classified into five classes: (a) features that capture the internal structure of the candidate argument, (b) features extracted from the argument context, (c) features that describe properties of the target predicate, (d) features generated from the predicate context, and (e) features that model the distance between the predicate and the argument. These five feature sets are listed in Tables 2, 3, 4, 5, and 6.

The *syntactic label* of the candidate constituent.
The constituent *head word*, *suffixes* of length 2, 3, and 4, *lemma*, and *POS tag*.
The constituent *content word*, *suffixes* of length 2, 3, and 4, *lemma*, *POS tag*, and *NE label*. Content words, which add informative lexicalized information different from the head word, were detected using the heuristics of (Surdeanu et al., 2003).
The *first and last constituent words* and their *POS tags*.
NE labels included in the candidate phrase.
Binary features to indicate the presence of *temporal cue words*, i.e. words that appear often in `AM-TMP` phrases in training.
For each TreeBank syntactic label we added a feature to indicate the *number of such labels* included in the candidate phrase.
The *sequence of syntactic labels* of the constituent immediate children.

Table 2: Argument structure features

The phrase *label, head word and POS tag* of the constituent parent, left sibling, and right sibling.

Table 3: Argument context features

2.3 Classifier

The classifiers used in this paper were developed using AdaBoost with confidence rated predictions (Schapire and Singer, 1999). AdaBoost combines many simple base classifiers or rules (in our case decision trees of depth 3) into a single strong classifier using a weighted-voted scheme. Each base classifier is learned sequentially from weighted examples and the weights are dynamically adjusted every learning iteration based on the behavior of the

The predicate *word* and *lemma*.
The predicate *voice*. We currently distinguish five voice types: active, passive, copulative, infinitive, and progressive.
A binary feature to indicate if the predicate is *frequent* - i.e. it appears more than twice in the training partition - or not.

Table 4: Predicate structure features

Sub-categorization rule, i.e. the phrase structure rule that expands the predicate immediate parent, e.g. NP → VBG NN NNS for the predicate in Figure 1 (b).

Table 5: Predicate context features

The *path* in the syntactic tree between the argument phrase and the predicate as a chain of syntactic labels along with the traversal direction (up or down).
The *length* of the above syntactic path.
The *number of clauses* (S* phrases) in the path.
The *number of verb phrases* (VP) in the path.
The *subsumption count*, i.e. the difference between the depths in the syntactic tree of the argument and predicate constituents. This value is 0 if the two phrases share the same parent.
The *governing category*, which indicates if NP arguments are dominated by a sentence (typical for subjects) or a verb phrase (typical for objects).
We *generalize* syntactic paths with more than 3 elements using two templates: (a) Arg ↑ Ancestor ↓ N_i ↓ Pred, where Arg is the argument label, Pred is the predicate label, Ancestor is the label of the common ancestor, and N_i is instantiated with all the labels between Pred and Ancestor in the full path; and (b) Arg ↑ N_i ↑ Ancestor ↓ Pred, where N_i is instantiated with all the labels between Arg and Ancestor in the full path.
The *surface distance* between the predicate and the argument phrases encoded as: the number of tokens, verb terminals (VB*), commas, and coordinations (CC) between the argument and predicate phrases, and a binary feature to indicate if the two constituents are adjacent.
A binary feature to indicate if the argument *starts with a predicate particle*, i.e. a token seen with the RP* POS tag and directly attached to the predicate in training.

Table 6: Predicate-argument distance features

previously learned rules.

We trained one-vs-all classifiers for the top 24 most common arguments in training (including R-A* and C-A*). For simplicity we do not label predicates. Following the strategy proposed by (Carreras et al., 2004) we select training examples (both positive and negative) only from: (a) the first S* phrase that includes the predicate, or (b) from phrases that appear to the left of the predicate in the sentence. More than 98% of the arguments fall into one of these classes.

At prediction time the classifiers are combined using a simple greedy technique that iteratively assigns to each predicate the argument classified with the highest confidence. For each predicate we consider as candidates all AM attributes, but only numbered attributes indicated in the corresponding PropBank frame.

2.4 Argument Expansion Heuristics

We address arguments that should map to more than one terminal phrase with the following post-processing heuristic: if an argument is mapped to one terminal phrase, its boundaries are extended to the right to include all terminal phrases that are not already labeled as other arguments for the same predicate. For example, after the system tags "consumer" as the beginning of an A1 argument in Figure 1, this heuristic extends the right boundary of the A1 argument to include the following terminal, "prices".

To handle inconsistencies in the treatment of quotes in parsing we added a second heuristic: arguments are expanded to include preceding/following quotes if the corresponding pairing quote is already included in the argument constituent.

3 Evaluation

3.1 Data

We trained our system using positive examples extracted from all training data available. Due to memory limitations on our development machines we used only the first 500,000 negative examples. In the experiments reported in this paper we used the syntactic trees generated by the Charniak parser. The results were evaluated for precision, recall, and F_1 using the scoring script provided by the task organizers.

3.2 Results and Discussion

Table 7 presents the results obtained by our system. On the WSJ data, our results surpass with almost 6% the results obtained by the best SRL system that used partial syntax in the CoNLL 2004 shared task evaluation (Hacioglu et al., 2004). Even though these numbers are not directly comparable (this year's shared task offers more training data), we consider these results encouraging given the simplicity of our system (we essentially model only one-to-one

	Precision	Recall	$F_{\beta=1}$
Development	79.14%	71.57%	75.17
Test WSJ	80.32%	72.95%	76.46
Test Brown	72.41%	59.67%	65.42
Test WSJ+Brown	79.35%	71.17%	75.04

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	80.32%	72.95%	76.46
A0	87.09%	85.21%	86.14
A1	79.80%	72.23%	75.83
A2	74.74%	58.38%	65.55
A3	83.04%	53.76%	65.26
A4	77.42%	70.59%	73.85
A5	0.00%	0.00%	0.00
AM-ADV	57.82%	46.05%	51.27
AM-CAU	49.38%	54.79%	51.95
AM-DIR	62.96%	40.00%	48.92
AM-DIS	72.19%	76.25%	74.16
AM-EXT	60.87%	43.75%	50.91
AM-LOC	64.19%	52.34%	57.66
AM-MNR	63.90%	44.77%	52.65
AM-MOD	98.09%	93.28%	95.63
AM-NEG	96.15%	97.83%	96.98
AM-PNC	55.22%	32.17%	40.66
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	79.17%	73.41%	76.18
R-A0	84.85%	87.50%	86.15
R-A1	75.00%	71.15%	73.03
R-A2	60.00%	37.50%	46.15
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	68.00%	80.95%	73.91
R-AM-MNR	30.00%	50.00%	37.50
R-AM-TMP	60.81%	86.54%	71.43
V	0.00%	0.00%	0.00

Table 7: Overall results (top) and detailed results on the WSJ test (bottom).

mappings between semantic arguments and syntactic constituents). Only 0.14% out of the 75.17% F measure obtained on the development partition are attributed to the argument expansion heuristics introduced in Section 2.4.

4 Conclusions

This paper describes a semantic role labeling system constructed on top of the complete syntactic analysis of text. We model semantic arguments that map into exactly one syntactic phrase (about 90% of all semantic arguments in the development set) using a rich set of lexical, syntactic, and semantic attributes. We trained AdaBoost one-versus-all classifiers for the 24 most common argument types. Arguments that map to more than one syntactic constituent are expanded with a simple heuristic in a post-processing step.

Our results surpass with almost 6% the results obtained by best SRL system that used partial syntax in the CoNLL 2004 shared task evaluation. Although the two evaluations are not directly comparable due to differences in training set size, the current results are encouraging given the simplicity of our proposed system.

5 Acknowledgements

This research has been partially funded by the European Union project "Computers in the Human Interaction Loop" (CHIL - IP506909). Mihai Surdeanu is a research fellow within the Ramón y Cajal program of the Spanish Ministry of Education and Science.

We would also like to thank Lluís Màrquez and Xavi Carreras for the help with the AdaBoost classifier, for providing the set of temporal cue words, and for the many motivating discussions.

References

X. Carreras and L. Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL 2004 Shared Task*.

X. Carreras, L. Màrquez, and G. Chrupała. 2004. Hierarchical recognition of propositional arguments with perceptrons. In *Proceedings of CoNLL 2004 Shared Task*.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD Dissertation, University of Pennsylvania.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3).

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2004. Semantic role labeling by tagging syntactic chunks. In *Proceedings of CoNLL 2004 Shared Task*.

S. Pradhan, K. Hacioglu, V. Krugler, W. Ward, J. H. Martin, and D. Jurafsky. 2005. Support vector learning for semantic argument classification. *To appear in Journal of Machine Learning*.

R. E. Schapire and Y. Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3).

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of ACL 2003*.

Joint Parsing and Semantic Role Labeling

Charles Sutton and Andrew McCallum
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
{casutton,mccallum}@cs.umass.edu

Abstract

A striking feature of human syntactic processing is that it is *context-dependent*, that is, it seems to take into account semantic information from the discourse context and world knowledge. In this paper, we attempt to use this insight to bridge the gap between SRL results from gold parses and from automatically-generated parses. To do this, we jointly perform parsing and semantic role labeling, using a probabilistic SRL system to rerank the results of a probabilistic parser. Our current results are negative, because a locally-trained SRL model can return inaccurate probability estimates.

1 Introduction

Although much effort has gone into developing statistical parsing models and they have improved steadily over the years, in many applications that use parse trees errors made by the parser are a major source of errors in the final output. A promising approach to this problem is to perform both parsing and the higher-level task in a single, *joint* probabilistic model. This not only allows uncertainty about the parser output to be carried upward, such as through an k-best list, but also allows information from higher-level processing to improve parsing. For example, Miller et al. (2000) showed that performing parsing and information extraction in a joint model improves performance on both tasks. In particular, one suspects that attachment decisions, which are both notoriously hard and extremely important for semantic analysis, could benefit greatly from input from higher-level semantic analysis.

The recent interest in semantic role labeling provides an opportunity to explore how higher-level semantic information can inform syntactic parsing. In previous work, it has been shown that SRL systems that use full parse information perform better than those that use shallow parse information, but that machine-generated parses still perform much worse than human-corrected gold parses.

The goal of this investigation is to narrow the gap between SRL results from gold parses and from automatic parses. We aim to do this by jointly performing parsing and semantic role labeling in a single probabilistic model. In both parsing and SRL, state-of-the-art systems are probabilistic; therefore, their predictions can be combined in a principled way by multiplying probabilities. In this paper, we rerank the k-best parse trees from a probabilistic parser using an SRL system. We compare two reranking approaches, one that linearly weights the log probabilities, and the other that learns a reranker over parse trees and SRL frames in the manner of Collins (2000).

Currently, neither method performs better than simply selecting the top predicted parse tree. We discuss some of the reasons for this; one reason being that the ranking over parse trees induced by the semantic role labeling score is unreliable, because the model is trained locally.

2 Base SRL System

Our approach to joint parsing and SRL begins with a base SRL system, which uses a standard architecture from the literature. Our base SRL system is a cascade of maximum-entropy classifiers which select the semantic argument label for each constituent of a full parse tree. As in other systems, we use three stages: pruning, identification, and classification. First, in *pruning*, we use a deterministic preprocessing procedure introduced by Xue and Palmer (2004) to prune many constituents which are almost certainly not arguments. Second, in *identification*, a binary MaxEnt classifier is used to prune remaining constituents which are predicted to be null with

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 225–228, Ann Arbor, June 2005.

Base features [GJ02]
Path to predicate
Constituent type
Head word
Position
Predicate
Head POS [SHWA03]
All conjunctions of above

Table 1: Features used in base identification classifier.

high probability. Finally, in *classification*, a multiclass MaxEnt classifier is used to predict the argument type of the remaining constituents. This classifer also has the option to output NULL.

It can happen that the returned semantic arguments overlap, because the local classifiers take no global constraints into account. This is undesirable, because no overlaps occur in the gold semantic annotations. We resolve overlaps using a simple recursive algorithm. For each parent node that overlaps with one of its descendents, we check which predicted probability is greater: that the parent has its locally-predicted argument label and all its descendants are null, or that the descendants have their optimal labeling, and the parent is null. This algorithm returns the non-overlapping assignment with globally highest confidence. Overlaps are uncommon, however; they occurred only 68 times on the 1346 sentences in the development set.

We train the classifiers on PropBank sections 02–21. If a true semantic argument fails to match any bracketing in the parse tree, then it is ignored. Both the identification and classification models are trained using gold parse trees. All of our features are standard features for this task that have been used in previous work, and are listed in Tables 1 and 2. We use the maximum-entropy implementation in the Mallet toolkit (McCallum, 2002) with a Gaussian prior on parameters.

3 Reranking Parse Trees Using SRL Information

Here we give the general framework for the reranking methods that we present in the next section. We write a joint probability model over semantic frames F and parse trees t given a sentence $\mathbf{x}$ as

$$p(F, t|\mathbf{x}) = p(F|t, \mathbf{x})p(t|\mathbf{x}), \quad (1)$$

where $p(t|\mathbf{x})$ is given by a standard probabilistic parsing model, and $p(F|t, \mathbf{x})$ is given by the baseline SRL model described previously.

Base features [GJ02]
Head word
Constituent type
Position
Predicate
Voice
Head POS [SHWA03]
From [PWHMJ04]
Parent Head POS
First word / POS
Last word / POS
Sibling constituent type / head word / head POS
Conjunctions [XP03]
Voice & Position
Predicate & Head word
Predicate & Constituent type

Table 2: Features used in baseline labeling classifier.

Parse Trees Used	SRL F1
Gold	77.1
1-best	63.9
Reranked by gold parse F1	68.1
Reranked by gold frame F1	74.2
Simple SRL combination ($\alpha = 0.5$)	56.9
Chosen using trained reranker	63.6

Table 3: Comparison of Overall SRL F1 on development set by the type of parse trees used.

In this paper, we choose (F^*, t^*) to approximately maximize the probability $p(F, t|\mathbf{x})$ using a reranking approach. To do the reranking, we generate a list of k-best parse trees for a sentence, and for each predicted tree, we predict the best frame using the base SRL model. This results in a list $\{(F^i, t^i)\}$ of parse tree / SRL frame pairs, from which the reranker chooses. Thus, our different reranking methods vary only in which parse tree is selected; given a parse tree, the frame is always chosen using the best prediction from the base model.

The k-best list of parses is generated using Dan Bikel's (2004) implementation of Michael Collins' parsing model. The parser is trained on sections 2–21 of the WSJ Treebank, which does not overlap with the development or test sets. The k-best list is generated in Bikel's implementation by essentially turning off dynamic programming and doing very aggressive beam search. We gather a maximum of 500 best parses, but the limit is not usually reached using feasible beam widths. The mean number of parses per sentence is 176.

4 Results and Discussion

In this section we present results on several reranking methods for joint parsing and semantic role la-

beling. Table 3 compares F1 on the development set of our different reranking methods. The first four rows in Table 3 are baseline systems. We present baselines using gold trees (row 1 in Table 3) and predicted trees (row 2). As shown in previous work, gold trees perform much better than predicted trees.

We also report two cheating baselines to explore the maximum possible performance of a reranking system. First, we report SRL performance of ceiling parse trees (row 3), i.e., if the parse tree from the k-best list is chosen to be closest to the gold tree. This is the best expected performance of a parse reranking approach that maximizes parse F1. Second, we report SRL performance where the parse tree is selected to maximize SRL F1, computing using the gold frame (row 4). There is a significant gap both between parse-F1-reranked trees and SRL-F1-reranked trees, which shows promise for joint reranking. However, the gap between SRL-F1-reranked trees and gold parse trees indicates that reranking of parse lists cannot by itself completely close the gap in SRL performance between gold and predicted parse trees.

4.1 Reranking based on score combination

Equation 1 suggests a straightforward method for reranking: simply pick the parse tree from the k-best list that maximizes $p(F, t|\mathbf{x})$, in other words, add the log probabilities from the parser and the base SRL system. More generally, we consider weighting the individual probabilities as

$$s(F, t) = p(F|t, \mathbf{x})^{1-\alpha} p(t|\mathbf{x})^{\alpha}. \quad (2)$$

Such a weighted combination is often used in the speech community to combine acoustic and language models.

This reranking method performs poorly, however. No choice of α performs better than $\alpha = 1$, i.e., choosing the 1-best predicted parse tree. Indeed, the more weight given to the SRL score, the worse the combined system performs. The problem is that often a bad parse tree has many nodes which are obviously not constituents: thus $p(F|t, \mathbf{x})$ for such a bad tree is very high, and therefore not reliable. As more weight is given to the SRL score, the unlabeled recall drops, from 55% when $\alpha = 0$ to 71% when $\alpha = 1$. Most of the decrease in F1 is due to the drop in unlabeled recall.

4.2 Training a reranker using global features

One potential solution to this problem is to add features of the entire frame, for example, to vote against predicted frames that are missing key arguments. But such features depend globally on the entire frame, and cannot be represented by local classifiers. One way to train these global features is to learn a linear classifier that selects a parse / frame pair from the ranked list, in the manner of Collins (2000). Reranking has previously been applied to semantic role labeling by Toutanova et al. (2005), from which we use several features. The difference between this paper and Toutanova et al. is that instead of reranking k-best SRL frames of a single parse tree, we are reranking 1-best SRL frames from the k-best parse trees.

Because of the the computational expense of training on k-best parse tree lists for each of 30,000 sentences, we train the reranker only on sections 15–18 of the Treebank (the same subset used in previous CoNLL competitions). We train the reranker using LogLoss, rather than the boosting loss used by Collins. We also restrict the reranker to consider only the top 25 parse trees.

This globally-trained reranker uses all of the features from the local model, and the following global features: (a) *sequence features*, i.e., the linear sequence of argument labels in the sentence (e.g. `A0_V_A1`), (b) the log probability of the parse tree, (c) *has-arg* features, that is, for each argument type a binary feature indicating whether it appears in the frame, (d) the conjunction of the predicate and has-arg feature, and (e) the number of nodes in the tree classified as each argument type.

The results of this system on the development set are given in Table 3 (row 6). Although this performs better than the score combination method, it is still no better than simply taking the 1-best parse tree. This may be due to the limited training set we used in the reranking model. A base SRL model trained only on sections 15–18 has 61.26 F1, so in comparison, reranking provides a modest improvement. This system is the one that we submitted as our official submission. The results on the test sets are given in Table 4.

5 Summing over parse trees

In this section, we sketch a different approach to joint SRL and parsing that does not use reranking at all. Maximizing over parse trees can mean that poor parse trees can be selected if their semantic labeling has an erroneously high score. But we are not actually interested in selecting a good parse tree; all we want is a good semantic frame. This means that we should select the semantic frame

	Precision	Recall	$F_{\beta=1}$
Development	64.43%	63.11%	63.76
Test WSJ	68.57%	64.99%	66.73
Test Brown	62.91%	54.85%	58.60
Test WSJ+Brown	67.86%	63.63%	65.68

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	68.57%	64.99%	66.73
A0	69.47%	74.35%	71.83
A1	66.90%	64.91%	65.89
A2	64.42%	61.17%	62.75
A3	62.14%	50.29%	55.59
A4	72.73%	70.59%	71.64
A5	50.00%	20.00%	28.57
AM-ADV	55.90%	49.60%	52.57
AM-CAU	76.60%	49.32%	60.00
AM-DIR	57.89%	38.82%	46.48
AM-DIS	79.73%	73.75%	76.62
AM-EXT	66.67%	43.75%	52.83
AM-LOC	50.26%	53.17%	51.67
AM-MNR	54.32%	51.16%	52.69
AM-MOD	98.50%	95.46%	96.96
AM-NEG	98.20%	94.78%	96.46
AM-PNC	46.08%	40.87%	43.32
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	72.15%	67.43%	69.71
R-A0	0.00%	0.00%	0.00
R-A1	0.00%	0.00%	0.00
R-A2	0.00%	0.00%	0.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	0.00%	0.00%	0.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-TMP	0.00%	0.00%	0.00
V	99.21%	86.24%	92.27

Table 4: Overall results (top) and detailed results on the WSJ test (bottom).

that maximizes the posterior probability: $p(F|\mathbf{x}) = \sum_t p(F|t,\mathbf{x})p(t|\mathbf{x})$. That is, we should be *summing* over the parse trees instead of maximizing over them. The practical advantage of this approach is that even if one seemingly-good parse tree does not have a constituent for a semantic argument, many other parse trees in the k-best list might, and all are considered when computing F^*. Also, no single parse tree need have constituents for all of F^*; because it sums over all parse trees, it can mix and match constituents between different trees. The optimal frame F^* can be computed by an $O(N^3)$ parsing algorithm if appropriate independence assumptions are made on $p(F|\mathbf{x})$. This requires designing an SRL model that is independent of the bracketing derived from any particular parse tree. Initial experiments performed poorly because the marginal model $p(F|\mathbf{x})$ was inadequate. Detailed exploration is left for future work.

6 Conclusion and Related Work

In this paper, we have considered several methods for reranking parse trees using information from semantic role labeling. So far, we have not been able to show improvement over selecting the 1-best parse tree. Gildea and Jurafsky (Gildea and Jurafsky, 2002) also report results on reranking parses using an SRL system, with negative results. In this paper, we confirm these results with a MaxEnt-trained SRL model, and we extend them to show that weighting the probabilities does not help either.

Our results with Collins-style reranking are too preliminary to draw definite conclusions, but the potential improvement does not appear to be great. In future work, we will explore the max-sum approach, which has promise to avoid the pitfalls of max-max reranking approaches.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval, in part by National Science Foundation under NSF grants #IIS-0326249 ond #IIS-0427594, and in part by the Defense Advanced Research Projec ts Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pages 175–182. Morgan Kaufmann, San Francisco, CA.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

Scott Miller, Heidi Fox, Lance A. Ramshaw, and Ralph M. Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *ANLP 2000*, pages 226–233.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *ACL-2003*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *ACL 2005*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*.

Applying spelling error correction techniques for improving semantic role labelling

Erik Tjong Kim Sang
Informatics Institute
University of Amsterdam, Kruislaan 403
NL-1098 SJ Amsterdam, The Netherlands
erikt@science.uva.nl

Sander Canisius, Antal van den Bosch, Toine Bogers
ILK / Computational Linguistics and AI
Tilburg University, P.O. Box 90153,
NL-5000 LE Tilburg, The Netherlands
{S.V.M.Canisius,Antal.vdnBosch,
A.M.Bogers}@uvt.nl

1 Introduction

This paper describes our approach to the CoNLL-2005 shared task: semantic role labelling. We do many of the obvious things that can be found in the other submissions as well. We use syntactic trees for deriving instances, partly at the constituent level and partly at the word level. On both levels we edit the data down to only the predicted positive cases of verb-constituent or verb-word pairs exhibiting a verb-argument relation, and we train two next-level classifiers that assign the appropriate labels to the positively classified cases. Each classifier is trained on data in which the features have been selected to optimize generalization performance on the particular task. We apply different machine learning algorithms and combine their predictions.

As a novel addition, we designed an automatically trained post-processing module that attempts to correct some of the errors made by the base system. To this purpose we borrowed Levenshtein-distance-based correction, a method from spelling error correction to repair mistakes in sequences of labels. We adapted the method to our needs and applied it for improving semantic role labelling output. This paper presents the results of our approach.

2 Data and features

The CoNLL-2005 shared task data sets provide sentences in which predicate–argument relations have been annotated, as well as a number of extra annotations like named entities and full syntactic parses (Carreras and Màrquez, 2005). We have used the parses for generating machine learning instances for pairs of predicates and syntactic phrases. In principle each phrase can have a relation with each verb in the same sentence. However, in order to keep the number of instances at a reasonable number, we have only built instances for verb–phrase pairs when the phrase parent is an ancestor of the verb (400,128 training instances). A reasonable number of arguments are individual words; these do not match with phrase boundaries. In order to be able to label these, we have also generated instances for all pairs of verbs and individual words using the same constraint (another 542,217 instances). The parent node constraint makes certain that embedded arguments, which do not occur in these data sets, cannot be predicted by our approach.

Instances which are associated with verb–argument pairs receive the label of the argument as class while others in principle receive a NULL class. In an estimated 10% of the cases, the phrase boundaries assigned by the parser are different from those in the argument annotation. In case of a mismatch, we have always used the argument label of the first word of a phrase as the class of the corresponding instance. By doing this we attempt to keep the positional information of the lost argument in the training data. Both the parser phrase boundary errors as well as the parent node constraint restrict the number of phrases we can identify. The maximum recall score attainable with our phrases is 84.64% for the development data set.

We have experimentally evaluated 30 features based on the previous work in semantic role labelling (Gildea and Jurafsky, 2002; Pradhan et al., 2004; Xue and Palmer, 2004):

- **Lexical features** (5): predicate (verb), first phrase word, last phrase word and words immediately before and after the phrase.
- **Syntactic features** (14): part-of-speech tags (POS) of: first phrase word, last phrase word,

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 229–232, Ann Arbor, June 2005.

word immediately before phrase and word immediately after phrase; syntactic paths from word to verb: all paths, only paths for words before verb and only paths for words after verb; phrase label, label of phrase parent, subcategorisation of verb parent, predicate frame from PropBank, voice, head preposition for prepositional phrases and same parents flag.
- **Semantic features** (2): named entity tag for first phrase word and last phrase word.
- **Positional features** (3): position of the phrase with respect to the verb: left/right, distance in words and distance in parent nodes.
- **Combination features** (6): predicate + phrase label, predicate + first phrase word, predicate + last phrase word, predicate + first phrase POS, predicate + last phrase POS and voice + left/right.

The output of two parsers was available. We have briefly experimented with the Collins parses including the available punctuation corrections but found that our approach reached a better performance with the Charniak parses. We report only on the results obtained with the Charniak parses.

3 Approach

This section gives a brief overview of the three main components of our approach: machine learning, automatic feature selection and post-processing by a novel procedure designed to clean up the classifier output by correcting obvious misclassifications.

3.1 Machine learning

The core machine learning technique employed, is memory-based learning, a supervised inductive algorithm for learning classification tasks based on the k-nn algorithm. We use the TiMBL system (Daelemans et al., 2003), version 5.0.0, patch-2 with uniform feature weighting and random tiebreaking (options: -w 0 -R 911). We have also evaluated two alternative learning techniques. First, Maximum Entropy Models, for which we employed Zhang Le's Maximum Entropy Toolkit, version 20041229 with default parameters. Second, Support Vector Machines for which we used Taku Kudo's YamCha (Kudo and Matsumoto, 2003), with one-versus-all voting and option -V which enabled us to ignore predicted classes with negative distances.

3.2 Feature selection

In previous research, we have found that memory-based learning is rather sensitive to the chosen features. In particular, irrelevant or redundant features may lead to reduced performance. In order to minimise the effects of this sensitivity, we have employed bi-directional hill-climbing (Caruana and Freitag, 1994) for finding the features that were most suited for this task. This process starts with an empty feature set, examines the effect of adding or removing one feature and then starts a new iteration with the set associated with the best performance.

3.3 Automatic post-processing

Certain misclassifications by the semantic role-labelling system described so far lead to unlikely and impossible relation assignments, such as assigning two indirect objects to a verb where only one is possible. Our proposed classifier has no mechanism to detect these errors. One solution is to devise a post-processing step that transforms the resulting role assignments until they meet certain basic constraints, such as the rule that each verb may have only single instances of the different roles assigned in one sentence (Van den Bosch et al., 2004).

We propose an alternative automatically-trained post-processing method which corrects unlikely role assignments either by deleting them or by replacing them with a more likely one. We do not do this by knowledge-based constraint satisfaction, but rather by adopting a method for error correction based on Levenshtein distance (Levenshtein, 1965), or edit distance, as used commonly in spelling error correction. Levenshtein distance is a dynamically computed distance between two strings, accounting for the number of deletions, insertions, and substitutions needed to transform the one string into the other. Levenshtein-based error correction typically matches a new, possibly incorrect, string to a trusted lexicon of assumedly correct strings, finds the lexicon string with the smallest Levenshtein distance to the new string, and replaces the new string with the lexicon string as its likely correction. We implemented a roughly similar procedure. First, we generated a lexicon of semantic role labelling patterns of A0–A5 arguments of verbs on the basis of the entire training corpus and the PropBank verb frames. This

lexicon contains entries such as *abandon* A0 V A1, and *categorize* A1 V A2 – a total of 43,033 variable-length role labelling patterns.

Next, given a new test sentence, we consider all of its verbs and their respective predicted role labellings, and compare each with the lexicon, searching the role labelling pattern with the same verb at the smallest Levenshtein distance (in case of an unknown verb we search in the entire lexicon). For example, in a test sentence the pattern *emphasize* A0 V A1 A0 is predicted. One closest lexicon item is found at Levenshtein distance 1, namely *emphasize* A0 V A1, representing a deletion of the final A0. We then use the nearest-neighbour pattern in the lexicon to correct the likely error, and apply all deletions and substitutions needed to correct the current pattern according to the nearest-neighbour pattern from the trusted lexicon. We do not apply insertions, since the post-processor module does not have the information to decide which constituent or word would receive the inserted label. In case of multiple possible deletions (e.g. in deleting one out of two A1s in *emphasize* A0 V A1 A1), we always delete the argument furthest from the verb.

4 Results

In order to perform the optimisation of the semantic role labelling process in a reasonable amount of time, we have divided it in four separate tasks: pruning the data for individual words and the data for phrases, and labelling of these two data sets. Pruning amounts to deciding which instances correspond with verb-argument pairs and which do not. This resulted in a considerable reduction of the two data sets: 47% for the phrase data and 80% for the word data. The remaining instances are assumed to define verb-argument pairs and the labelling tasks assign labels to them. We have performed a separate feature selection process in combination with the memory-based learner for each of the four tasks. First we selected the best feature set based on task accuracy. As soon as a working module for each of the tasks was available, we performed an extra feature selection process for each of the modules, optimising overall system $F_{\beta=1}$ while keeping the other three modules fixed.

The effect of the features on the overall performance can be found in Table 1. One feature (syntactic path) was selected in all four tasks but in general different features were required for optimal performance in the four tasks. Changing the feature set had the largest effect when labelling the phrase data. We have applied the two other learners, Maximum Entropy Models and Support Vector Machines to the two labelling tasks, while using the same features as the memory-based learner. The performance of the three systems on the development data can be found in Table 3. Since the systems performed differently we have also evaluated the performance of a combined system which always chose the majority class assigned to an instance and the class of the strongest system (SVM) in case of a three-way tie. The combined system performed slightly better than the best

	Words		Phrases	
Features	prune	label	prune	label
predicate	-0.04	**+0.05**	-0.25	-0.52
first word	**+0.38**	**+0.16**	-0.17	**+1.14**
last word	–	–	-0.01	**+1.12**
previous word	-0.06	**+0.02**	-0.05	**+0.74**
next word	-0.04	-0.08	**+0.44**	-0.16
part-of-speech first word	-0.01	-0.02	-0.07	-0.11
part-of-speech last word	–	–	-0.14	-0.45
previous part-of-speech	-0.12	-0.06	**+0.22**	-1.14
next part-of-speech	-0.08	-0.12	-0.01	-0.21
all paths	**+0.42**	**+0.10**	**+0.84**	**+0.75**
path before verb	+0.00	-0.02	+0.00	**+0.27**
path after verb	-0.01	-0.01	-0.01	-0.06
phrase label	-0.01	-0.02	**+0.13**	-0.02
parent label	**+0.03**	-0.02	-0.03	+0.00
voice	+0.02	-0.04	-0.04	**+1.85**
subcategorisation	-0.01	+0.00	-0.02	+0.03
PropBank frame	-0.12	-0.03	-0.16	**+1.04**
PP head	+0.00	+0.00	-0.06	**+0.08**
same parents	-0.02	-0.01	**+0.03**	-0.05
named entity first word	+0.00	+0.00	+0.05	-0.11
named entity last word	–	–	-0.04	-0.12
absolute position	**+0.00**	+0.00	+0.00	-0.02
distance in words	**+0.34**	**+0.04**	**+0.16**	-0.96
distance in parents	-0.02	-0.02	**+0.06**	-0.04
predicate + label	-0.05	-0.07	-0.22	-0.47
predicate + first word	-0.05	+0.00	**+0.13**	**+0.97**
predicate + last word	–	–	-0.03	**+0.08**
predicate + first POS	-0.05	-0.06	-0.20	-0.50
predicate + last POS	–	–	-0.13	-0.40
voice + position	+0.02	-0.04	-0.05	-0.04

Table 1: Effect of adding a feature to the best feature sets when memory-based learning is applied to the development set (overall $F_{\beta=1}$). The process consisted of four tasks: pruning data sets for individual words and phrases, and labelling these two data sets. Selected features are shown in **bold**. Unfortunately, we have not been able to use all promising features.

	Precision	Recall	$F_{\beta=1}$
Development	76.79%	70.01%	73.24
Test WSJ	79.03%	72.03%	75.37
Test Brown	70.45%	60.13%	64.88
Test WSJ+Brown	77.94%	70.44%	74.00

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	79.03%	72.03%	75.37
A0	85.65%	81.73%	83.64
A1	76.97%	71.89%	74.34
A2	71.07%	58.20%	63.99
A3	69.29%	50.87%	58.67
A4	75.56%	66.67%	70.83
A5	100.00%	40.00%	57.14
AM-ADV	64.36%	51.38%	57.14
AM-CAU	75.56%	46.58%	57.63
AM-DIR	48.98%	28.24%	35.82
AM-DIS	81.88%	79.06%	80.45
AM-EXT	87.50%	43.75%	58.33
AM-LOC	62.50%	50.96%	56.15
AM-MNR	64.52%	52.33%	57.78
AM-MOD	96.76%	97.64%	97.20
AM-NEG	97.38%	96.96%	97.17
AM-PNC	45.98%	34.78%	39.60
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	80.52%	70.75%	75.32
R-A0	81.47%	84.38%	82.89
R-A1	74.00%	71.15%	72.55
R-A2	60.00%	37.50%	46.15
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	100.00%	100.00%	100.00
R-AM-LOC	86.67%	61.90%	72.22
R-AM-MNR	33.33%	33.33%	33.33
R-AM-TMP	64.41%	73.08%	68.47
V	97.36%	97.36%	97.36

Table 2: Overall results (top) and detailed results on the WSJ test (bottom).

individual system.

5 Conclusion

We have presented a machine learning approach to semantic role labelling based on full parses. We have split the process in four separate tasks: pruning the data bases of word-based and phrase-based examples down to only the positive verb-argument cases, and labelling the two positively classified data sets. A novel automatic post-processing procedure based on spelling correction, comparing to a trusted lexicon of verb-argument patterns from the training material, was able to achieve a performance increase by correcting unlikely role assignments.

Learning algorithm	Precision	Recall	$F_{\beta=1}$
without post-processing:			
Maximum Entropy Models	70.78%	70.03%	70.40
Memory-Based Learning	70.70%	69.85%	70.27
Support Vector Machines	75.07%	69.15%	71.98
including post-processing:			
Maximum Entropy Models	74.06%	69.84%	71.89
Memory-Based Learning	73.84%	69.88%	71.80
Support Vector Machines	77.75%	69.11%	73.17
Combination	76.79%	70.01%	73.24

Table 3: Effect of the choice of machine learning algorithm, the application of Levenshtein-distance-based post-processing and the use of system combination on the performance obtained for the development data set.

Acknowledgements

This research was funded by NWO, the Netherlands Organisation for Scientific Research, and by Senter-Novem IOP-MMI.

References

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL-2005*. Ann Arbor, MI, USA.

R. Caruana and D. Freitag. 1994. Greedy attribute selection. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 28–36, New Brunswick, NJ, USA. Morgan Kaufman.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2003. TiMBL: Tilburg memory based learner, version 5.0, reference guide. ILK Technical Report 03-10, Tilburg University.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

T. Kudo and Y. Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL-2003*. Sapporo, Japan.

V. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848.

S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the HLT/NAACL 2004*. Boston, MA.

A. van den Bosch, S. Canisius, W. Daelemans, I Hendrickx, and E. Tjong Kim Sang. 2004. Memory-based semantic role labeling: Optimizing features, algorithm, and output. In *Proceedings of the CoNLL-2004*, Boston, MA, USA.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP-2004*. Barcelona, Spain.

Exploiting Full Parsing Information to Label Semantic Roles Using an Ensemble of ME and SVM via Integer Linear Programming

Tzong-Han Tsai, Chia-Wei Wu, Yu-Chun Lin, Wen-Lian Hsu
Institute of Information Science
Academia Sinica
Taipei 115, Taiwan
{thtsai, cwwu, sbb, hsu}@iis.sinica.edu.tw

Abstract

In this paper, we propose a method that exploits full parsing information by representing it as features of argument classification models and as constraints in integer linear learning programs. In addition, to take advantage of SVM-based and Maximum Entropy-based argument classification models, we incorporate their scoring matrices, and use the combined matrix in the above-mentioned integer linear programs. The experimental results show that full parsing information not only increases the F-score of argument classification models by 0.7%, but also effectively removes all labeling inconsistencies, which increases the F-score by 0.64%. The ensemble of SVM and ME also boosts the F-score by 0.77%. Our system achieves an F-score of 76.53% in the development set and 76.38% in Test WSJ.

1 Introduction

The Semantic Role Labeling problem can be formulated as a sentence tagging problem. A sentence can be represented as a sequence of words, as phrases (chunks), or as a parsing tree. The basic units of a sentence are words, phrases, and constituents in these representations, respectively.. Pradhan et al. (2004) established that Constituent-by-Constituent (C-by-C) is better than Phrase-by-Phrase (P-by-P), which is better than Word-by-Word (W-by-W). This is probably because the boundaries of the constituents coincide with the arguments; therefore, C-by-C has the highest argument identification F-score among the three approaches.

In addition, a full parsing tree also provides richer syntactic information than a sequence of chunks or words. Pradhan et al. (2004) compared the seven most common features as well as several features related to the target constituent's parent and sibling constituents. Their experimental results show that using other constituents' information increases the F-score by 6%. Punyakanok et al. (2004) represent full parsing information as constraints in integer linear programs. Their experimental results show that using such information increases the argument classification accuracy by 1%.

In this paper, we not only add more full parsing features to argument classification models, but also represent full parsing information as constraints in integer linear programs (ILP) to resolve label inconsistencies. We also build an ensemble of two argument classification models: Maximum Entropy and SVM by combining their argument classification results and applying them to the above-mentioned ILPs.

2 System Architecture

Our SRL system is comprised of four stages: *pruning*, *argument classification*, *classification model incorporation*, and *integer linear programming*. This section describes how we build these stages, including the features used in training the argument classification models.

2.1 Pruning

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 233–236, Ann Arbor, June 2005.

When the full parsing tree of a sentence is available, only the constituents in the tree are considered as argument candidates. In CoNLL-2005, full parsing trees are provided by two full parsers: the Collins parser (Collins, 1999) and the Charniak parser (Charniak, 2000). According to Punyakanok et al. (2005), the boundary agreement of Charniak is higher than that of Collins; therefore, we choose the Charniak parser's results. However, there are two million nodes on the full parsing trees in the training corpus, which makes the training time of machine learning algorithms extremely long. Besides, noisy information from unrelated parts of a sentence could also affect the training of machine learning models. Therefore, our system exploits the heuristic rules introduced by Xue and Palmer (2004) to filter out simple constituents that are unlikely to be arguments. Applying pruning heuristics to the output of Charniak's parser effectively eliminates 61% of the training data and 61.3% of the development data, while still achieves 93% and 85.5% coverage of the correct arguments in the training and development sets, respectively.

2.2 Argument Classification

This stage assigns the final labels to the candidates derived in Section 2.1. A multi-class classifier is trained to classify the types of the arguments supplied by the pruning stage. In addition, to reduce the number of excess candidates mistakenly output by the previous stage, these candidates can be labeled as null (meaning "not an argument"). The features used in this stage are as follows.

Basic Features

- **Predicate** – The predicate lemma.
- **Path** – The syntactic path through the parsing tree from the parse constituent being classified to the predicate.
- **Constituent Type**
- **Position** – Whether the phrase is located before or after the predicate.
- **Voice** – passive: if the predicate has a POS tag VBN, and its chunk is not a VP, or it is preceded by a form of "to be" or "to get" within its chunk; otherwise, it is active.
- **Head Word** – calculated using the head word table described by Collins (1999).
- **Head POS** – The POS of the Head Word.
- **Sub-categorization** – The phrase structure rule that expands the predicate's parent node in the parsing tree.
- **First and Last Word/POS**
- **Named Entities** – LOC, ORG, PER, and MISC.
- **Level** – The level in the parsing tree.

Combination Features

- **Predicate Distance Combination**
- **Predicate Phrase Type Combination**
- **Head Word and Predicate Combination**
- **Voice Position Combination**

Context Features

- **Context Word/POS** – The two words preceding and the two words following the target phrase, as well as their corresponding POSs.
- **Context Chunk Type** – The two chunks preceding and the two chunks following the target phrase.

Full Parsing Features

We believe that information from related constituents in the full parsing tree helps in labeling the target constituent. Denote the target constituent by *t*. The following features are the most common baseline features of *t*'s parent and sibling constituents. For example, Parent/ Left Sibling/ Right Sibling Path denotes *t*'s parents', left sibling's, and right sibling's Path features.

- **Parent / Left Sibling / Right Sibling Path**
- **Parent / Left Sibling / Right Sibling Constituent Type**
- **Parent / Left Sibling / Right Sibling Position**
- **Parent / Left Sibling / Right Sibling Head Word**
- **Parent / Left Sibling / Right Sibling Head POS**
- **Head of PP parent** – If the parent is a PP, then the head of this PP is also used as a feature.

Argument Classification Models

We use all the features of the SVM-based and ME-based argument classification models. All SVM classifiers are realized using SVM-Light with a polynomial kernel of degree 2. The ME-based model is implemented based on Zhang's MaxEnt toolkit[1] and L-BFGS (Nocedal and Wright, 1999) method to perform parameter estimation.

2.3 Classification Model Incorporation

We now explain how we incorporate the SVM-based and ME-based argument classification models. After argument classification, we acquire two scoring matrices, $\mathbf{P}_{ME}$ and $\mathbf{P}_{SVM}$, respectively. Incorporation of these two models is realized by weighted summation of $\mathbf{P}_{ME}$ and $\mathbf{P}_{SVM}$ as follows:

$$\mathbf{P}' = w_{ME}\mathbf{P}_{ME} + w_{SVM}\mathbf{P}_{SVM}$$

We use $\mathbf{P}'$ for the objective coefficients of the ILP described in Section 2.4.

2.4 Integer Linear Programming (ILP)

To represent full parsing information as features, there are still several syntactic constraints on a parsing tree in the SRL problem. For example, on a path of the parsing tree, there can be only one constituent annotated as a non-null argument. However, it is difficult to encode this constraint in the argument classification models. Therefore, we apply integer linear programming to resolve inconsistencies produced in the argument classification stage.

According to Punyakanok et al. (2004), given a set of constituents, **S**, and a set of semantic role labels, **A**, the SRL problem can be formulated as an ILP as follows:

Let z_{ia} be the indicator variable that represents whether or not an argument, a, is assigned to any $S_i \in \mathbf{S}$; and let $p_{ia} = \text{score}(S_i = a)$. The scoring matrix **P** composed of all p_{ia} is calculated by the argument classification models. The goal of this ILP is to find a set of assignments for all z_{ia} that maximizes the following function:

$$\sum_{S_i \in \mathbf{S}} \sum_{a \in \mathbf{A}} p_{ia} z_{ia} .$$

Each $S_i \in \mathbf{S}$ should have one of these argument types, or no type (null). Therefore, we have

$$\sum_{a \in \mathbf{A}} z_{ia} = 1 .$$

Next, we show how to transform the constraints in the filter function into linear equalities or inequalities, and use them in this ILP.

Constraint I: No overlapping or embedding

For arguments $S_{j_1}, \ldots, S_{j_k}$ on the same path in a full parsing tree, only one argument can be assigned to an argument type. Thus, at least $k - 1$ arguments will be *null*, which is represented by ϕ in the following linear equality:

$$\sum_{i=1}^{k} z_{j_i \phi} \geq k - 1 .$$

Constraint II: No duplicate argument classes

Within the same sentence, A0-A5 cannot appear more than once. The inequality for A0 is therefore:

$$\sum_{i=1}^{k} z_{i\text{A0}} \leq 1 .$$

Constraint III: R-XXX arguments

The linear inequalities that represent A0 and its reference type R-A0 are:

$$\forall m \in \{1,\ldots,M\} : \sum_{i=1}^{k} z_{i\text{A0}} \geq z_{m\text{R-A0}} .$$

Constraint IV: C-XXX arguments

The continued argument XXX has to occur before C-XXX. The linear inequalities for A0 are:

$$\forall m \in \{2,\ldots,M\} : \sum_{i=1}^{m-1} z_{j_i\text{A0}} \geq z_{m\text{C-A0}} .$$

Constraint V: Illegal arguments

For each verb, we look up its allowed roles. This constraint is represented by summing all the corresponding indicator variables to 0.

3 Experiment Results

3.1 Data and Evaluation Metrics

The data, which is part of the PropBank corpus, consists of sections from the Wall Street Journal part of the Penn Treebank. All experiments were carried out using Section 2 to Section 21 for training, Section 24 for development, and Section 23 for testing. Unlike CoNLL-2004, part of the Brown corpus is also included in the test set.

3.2 Results

Table 1 shows that our system makes little difference to the development set and Test WSJ. However, due to the intrinsic difference between the WSJ and Brown corpora, our system performs better on Test WSJ than on Test Brown.

[1] http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

	Precision	Recall	$F_{\beta=1}$
Development	81.13%	72.42%	76.53
Test WSJ	82.77%	70.90%	76.38
Test Brown	73.21%	59.49%	65.64
Test WSJ+Brown	81.55%	69.37%	74.97

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	82.77%	70.90%	76.38
A0	88.25%	84.93%	86.56
A1	82.21%	72.21%	76.89
A2	74.68%	52.34%	61.55
A3	78.30%	47.98%	59.50
A4	84.29%	57.84%	68.60
A5	100.00%	60.00%	75.00
AM-ADV	64.19%	47.83%	54.81
AM-CAU	70.00%	38.36%	49.56
AM-DIR	38.20%	40.00%	39.08
AM-DIS	83.33%	71.88%	77.18
AM-EXT	86.67%	40.62%	55.32
AM-LOC	63.71%	41.60%	50.33
AM-MNR	63.36%	48.26%	54.79
AM-MOD	98.00%	97.64%	97.82
AM-NEG	99.53%	92.61%	95.95
AM-PNC	44.44%	17.39%	25.00
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	83.21%	61.09%	70.45
R-A0	91.08%	86.61%	88.79
R-A1	79.49%	79.49%	79.49
R-A2	87.50%	43.75%	58.33
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	92.31%	57.14%	70.59
R-AM-MNR	25.00%	16.67%	20.00
R-AM-TMP	72.73%	61.54%	66.67
V	97.32%	97.32%	97.32

Table 1. Overall results (top) and detailed results on the WSJ test (bottom).

	Precision	Recall	$F_{\beta=1}$
ME w/o parsing	77.28%	70.55%	73.76%
ME	78.19%	71.08%	74.46%
ME with ILP	79.57%	71.11%	75.10%
SVM	79.88%	72.03%	75.76%
Hybrid	**81.13%**	**72.42%**	**76.53%**

Table 2. Results of all configurations on the development set.

From Table 2, we can see that the model with full parsing features outperforms the model without the features in all three performance matrices. After applying ILP, the performance is improved further. We also observe that SVM slightly outperforms ME. However, the hybrid argument classification model achieves the best results in all three metrics.

4 Conclusion

In this paper, we add more full parsing features to argument classification models, and represent full parsing information as constraints in ILPs to resolve labeling inconsistencies. We also integrate two argument classification models, ME and SVM, by combining their argument classification results and applying them to the above-mentioned ILPs. The results show full parsing information increases the total F-score by 1.34%. The ensemble of SVM and ME also boosts the F-score by 0.77%. Finally, our system achieves an F-score of 76.53% in the development set and 76.38% in Test WSJ.

Acknowledgement

We are indebted to Wen Shong Lin and Prof. Fu Chang for their invaluable advice in data pruning, which greatly speeds up the training of our machine learning models.

References

X. Carreras and L. Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of the CoNLL-2005.*

E. Charniak. 2000. A Maximum-Entropy-Inspired Parser. *Proceedings of the NAACL-2000*.

M. J. Collins. 1999. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania.

J. Nocedal and S. J. Wright. 1999. *Numerical Optimization*, Springer.

S. Pradhan, K. Hacioglu, V. Kruglery, W. Ward,J. H. Martin, and D. Jurafsky. 2004. Support Vector Learning for Semantic Argument Classification. *Journal of Machine Learning*.

V. Punyakanok, D. Roth, and W. Yih. 2005. The Necessity of Syntactic Parsing for Semantic Role Labeling. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05).*

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04).*

N. Xue and M. Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of the EMNLP 2004*.

The Integration of Syntactic Parsing and Semantic Role Labeling

Szu-ting Yi
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104 USA
szuting@linc.cis.upenn.edu

Martha Palmer
University of Pennsylvania
3330 Walnut Street
Philadelphia, PA 19104 USA
mpalmer@linc.cis.upenn.edu

Abstract

This paper describes a system for the CoNLL-2005 Shared Task on Semantic Role Labeling. We trained two parsers with the training corpus in which the semantic argument information is attached to the constituent labels, we then used the resulting parse trees as the input of the pipelined SRL system. We present our results of combining the output of various SRL systems using different parsers.

1 Introduction

Semantic parsing, identifying and classifying the semantic entities in context and the relations between them, potentially has great impact on its downstream applications, such as text summarization, question answering, and machine translation. As a result, semantic parsing could be an important intermediate step for natural language comprehension. In this paper, we investigate the task of Semantic Role Labeling (SRL): Given a verb in a sentence, the goal is to locate the constituents which are arguments of the verb, and assign them appropriate semantic roles, such as, Agent, Patient, and Theme.

Previous SRL systems have explored the effects of using different lexical features, and experimented on different machine learning algorithms. (Gildea and Palmer, 2002; Pradhan et al., 2005; Punyakanok et al., 2004) However, these SRL systems generally extract features from sentences processed by a syntactic parser or other shallow parsing components, such as a chunker and a clause identifier. As a result, the performance of the SRL systems relies heavily on those syntax-analysis tools.

In order to improve the fundamental performance of an SRL system, we trained parsers with training data containing not only syntactic constituent information but also semantic argument information. The new parsers generate more correct constituents than that trained on pure syntactic information. Because the new parser generate different constituents than a pure syntactic parser, we also explore the possibility of combining the output of several parsers with the help of a voting post-processing component.

This paper is organized as follows: Section 2 demonstrates the components of our SRL system. We elaborate the importance of training a new parser and outline our approach in Section 3 and Section 4. Finally, Section 5 reports and discusses the results.

2 Semantic Role Labeling: the Architecture

Our SRL system has 5 phases: Parsing, Pruning, Argument Identification, Argument Classification, and Post Processing. The Argument Identification and Classification components are trained with Sec 02-21 of the Penn Treebank corpus.

2.1 Parsing

Previous SRL systems usually use a pure syntactic parser, such as (Charniak, 2000; Collins, 1999), to retrieve possible constituents. Once the boundary of a constituent is defined, there is no way to change it in later phases. Therefore the quality of the syntactic parser has a major impact on the final per-

Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL),
pages 237–240, Ann Arbor, June 2005.

formance of an SRL system, and the percentage of correct constituents that is generated by the syntactic parser also defines the recall upper bound of an SRL system. In order to attack this problem, in addition to Charniak's parser (Charniak, 2000), our system combine two parser which are trained on both syntactic constituent information and semantic argument information. (See Section 3)

2.2 Pruning

Given a parse tree, a pruning component filters out the constituents which are unlikely to be semantic arguments in order to facilitate the training of the Argument Identification component. Our system uses the heuristic rules introduced by (Xue and Palmer, 2004). The heuristics first spot the verb and then extract all the sister nodes along the verb spine of the parse tree. We expand the coverage by also extracting all the immediate children of an S, ADVP, PP and NP node. This stage generally prunes off about 80% of the constituents given by a parser. For our newly trained parsers, we also extract constituents which have a secondary constituent label indicating the constituent in question is an argument.

2.3 Argument Identification and Classification

We have as our Argument Identification component a binary maximum-entropy classifier to determine whether a constituent is an argument or not. If a constituent is tagged as an argument, the Argument Classification component, which is a multi-class maximum-entropy classifier, would assign it a semantic role. The implementation of both the Argument Identification and Classification components makes use of the Mallet package[1].

The lexical features we use to train these two components are taken from (Xue and Palmer, 2004).

We trained the Argument Identification component with the following single features: the **path** from the constituent to the verb, the **head word** of the constituent and its **POS tag**, and the **distance** between the verb and the constituent, and **feature combinations**: the verb and the phrasal type of the constituent, the verb and the head word of the constituent. If the parent node of the constituent is a PP node, then we also include the head word of the PP node and the feature combination of the verb and the head word of the PP node.

In addition to the features listed above, the Argument Classification component also contains the following features: the **verb**, the first and the last **content word** of the constituent, the **phrasal type** of the left sibling and the parent node, **voice** (passive or active), **position** of the constituent relative to the verb, the **subcategorization frame**, and the **syntactic frame** which describes the sequential pattern of the noun phrases and the verb in the sentence.

2.4 Post Processing

The post processing component merges adjacent discontinuous arguments and marks the R-arguments based on the content word and phrase type of the argument. Also it filters out arguments according to the following constraints:

1. There are no overlapping arguments.
2. There are no repeating core arguments.

In order to combine the different systems, we also include a voting scheme. The algorithm is straightforward: Suppose there are N participating systems, we pick arguments with N votes, N-1 votes ..., and finally 1 vote. The way to break a tie is based on the confidence level of the argument given by the system. Whenever we pick an argument, we need to check whether this argument conflicts with previously selected arguments based on the constraints described above.

3 Training a Parser with Semantic Argument Information

A good start is always important, especially for a successful SRL system. Instead of passively accepting candidate constituents from the upstream syntactic parser, an SRL system needs to interact with the parser in order to obtain improved performance. This motivated our first attempt which is to integrate syntactic parsing and semantic parsing as a single step, and hopefully as a result we would be able to discard the SRL pipeline. The idea is to augment the Penn Treebank (Marcus et al., 1994) constituent labels with the semantic role labels from the PropBank (Palmer et al., 2005), and generate a rich training corpus. For example, if an *NP* is also an ar-

[1] http://mallet.cs.umass.edu

gument *ARG0* of a verb in the given sentence, we change the constituent label *NP* into *NP-ARG0*. A parser therefore is trained on this new corpus and should be able to serve as an SRL system at the same time as predicting a parse.

However, this ideal approach is not feasible. Given the fact that there are many different semantic role labels and the same constituent can be different arguments of different verbs in the same sentence, the number of constituent labels will soon grow out of control and make the parser training computationally infeasible. Not to mention that anchor verb information has not yet been added to the constituent label, and general data sparseness. As a compromise, we decided to integrate only Argument Identification with syntactic parsing. We generated the training corpus by simply marking the constituents which are also semantic arguments.

4 Parsing Experiments

We trained a maximum-entropy parser based on (Ratnaparkhi, 1999) using the OpenNLP package [2]. We started our experiments with this specific parsing implementation because of its excellent flexibility that allows us to test different features. Besides, this parser contains four clear parse tree building stages: TAG, CHUNK, BUILD, and CHECK. This parsing structure offers us an isolated working environment for each stage that helps us confine necessary implementation modifications and trace down implementation errors.

4.1 Data Preparation

Following standard practice, we use Sec 02-21 of the Penn Treebank and the PropBank as our training corpus. The constituent labels defined in the Penn Treebank consist of a primary label and several secondary labels. A primary label represents the major syntactic function carried by the constituent, for instance, NP indicates a noun phrase and PP indicates a prepositional phrase. A secondary label, starting with "-", represents either a grammatical function of a constituent or a semantic function of an adjunct. For example, NP-SBJ means the noun phrase is a surface subject of the sentence; PP-LOC means the prepositional phrase is a location. Although the secondary labels give us much to encourage information, because of data sparseness problem and training efficiency, we stripped off all the secondary labels from the Penn Treebank.

After stripping off the secondary labels from the Penn Treebank, we augment the constituent labels with the semantic argument information from the PropBank. We adopted four different labels, -AN, -ANC, -AM, and -AMC. If the constituent in the Penn Treebank is a core argument, which means the constituent has one of the labels of ARG0-5 and ARGA in the PropBank, we attach -AN to the constituent label. The label -ANC means the constituent is a discontinuous core argument. Similarly, -AM indicates an adjunct-like argument, ARGM, and -AMC indicates a discontinuous ARM.

For example, the sentence from Sec 02, *[ARG0 The luxury auto maker] [ARGM-TMP last year] sold [ARG1 1,214 cars] [ARGM-LOC in the U.S.]*, would appear in the following format in our training corpus: *(S (NP-AN (DT The) (NN luxury) (NN auto) (NN maker)) (NP-AM (JJ last) (NN year)) (VP (VBD sold) (NP-AN (CD 1,214) (NNS cars)) (PP -AM (IN in) (NP (DT the) (NNP U.S.)))))*

4.2 The 2 Different Parsers

Since the core arguments and the ARGMs in the PropBank loosely correspond to the complements and adjuncts in the linguistics literature, we are interested in investigating their individual effect on parsing performance. We trained two parsers. An **AN**-parser was trained on the Penn Treebank corpus augmented with two semantic argument labels: -AN, and -ANC. Another **AM**-parser was trained on labels -AM, and -AMC.

5 Results and Discussion

Table 1 shows the results after combining various SRL systems using different parsers. In order to explore the effects of combining, we include the overall performance on the development dataset of individual SRL systems in Table 2.

The performance of Semantic Role Labeling (SRL) is determined by the quality of the syntactic information provided to the system. In this paper, we investigate that for the SRL task whether it is more suitable to use a parser trained with data con-

[2] http://sourceforge.net/projects/opennlp/

	Precision	Recall	$F_{\beta=1}$
Development	75.70%	69.99%	72.73
Test WSJ	77.51%	72.97%	75.17
Test Brown	67.88%	59.03%	63.14
Test WSJ+Brown	76.31%	71.10%	73.61

Test WSJ	Precision	Recall	$F_{\beta=1}$
Overall	77.51%	72.97%	75.17
A0	85.14%	77.32%	81.04
A1	77.61%	75.16%	76.37
A2	68.18%	62.16%	65.03
A3	66.91%	52.60%	58.90
A4	77.08%	72.55%	74.75
A5	100.00%	40.00%	57.14
AM-ADV	59.73%	51.58%	55.36
AM-CAU	67.86%	52.05%	58.91
AM-DIR	65.67%	51.76%	57.89
AM-DIS	80.39%	76.88%	78.59
AM-EXT	78.95%	46.88%	58.82
AM-LOC	57.43%	55.37%	56.38
AM-MNR	54.37%	56.10%	55.22
AM-MOD	96.64%	94.01%	95.31
AM-NEG	96.88%	94.35%	95.59
AM-PNC	41.38%	41.74%	41.56
AM-PRD	50.00%	20.00%	28.57
AM-REC	0.00%	0.00%	0.00
AM-TMP	77.13%	74.15%	75.61
R-A0	86.82%	85.27%	86.04
R-A1	67.72%	82.05%	74.20
R-A2	46.15%	37.50%	41.38
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	0.00%	0.00%	0.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	100.00%	42.86%	60.00
R-AM-MNR	33.33%	33.33%	33.33
R-AM-TMP	78.57%	63.46%	70.21
R-C-A1	0.00%	0.00%	0.00
V	97.35%	95.54%	96.44

Table 1: Overall results (top) and detailed results on the WSJ test (bottom).

taining both syntactic bracketing and semantic argument boundary information than a pure syntactic one.

The results of the SRL systems using the AM- or AN- parsers are not significantly better than that using the Charniak's parser. This might due to the simple training mechanism of the base parsing algorithm which the AM- and AN- parsers exploit. It also suggests our future work to apply the approach to more sophisticated parsing frameworks. By then, We show that we can boost the final performance by combining different SRL systems using different parsers, given that the combination algorithm is capable of maintaining the quality of the final arguments.

	Precision	Recall	$F_{\beta=1}$
AN-parser	71.31%	63.68%	67.28
AM-parser	74.09%	65.11%	69.31
Charniak	76.31%	64.62%	69.98
All 3 combined	75.70%	69.99%	72.73

Table 2: Overall results on the development set of individual SRL systems.

6 Acknowledgments

We thank Tom Morton for providing detailed explanation for any of our parsing related inquiries.

References

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL-2000*.

Michael Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. *PhD Dissertation*, University of Pennsylvania.

Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of ACL* 2002, Philadelphia, USA.

Mitchell Marcus, Grace Kim, Mary AnnMarcinkiewicz, et al. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of ARPA Speech and Natural Language Workshop*.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1).

Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J., and Jurafsky, D. 2005. Support Vector Learning for Semantic Argument Classification. To appear in *Machine Learning* journal, Special issue on Speech and Natural Language Processing.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of COLING*.

Adwait Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning*, 34, 151–175.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of EMNLP*.

Author Index