2006

COLING • ACL

COLING • ACL 2006

21st International Conference on
Computational Linguistics
and
44th Annual Meeting of the
Association for Computational Linguistics

Proceedings of the Student Research Workshop

20 July 2006
Sydney, Australia

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
`acl@aclweb.org`

Table of Contents

Preface

Welcome to the COLING/ACL 2006 Student Research Workshop! The Student Research Workshop is now an established tradition at ACL conferences. This year, we received 40 submissions ranging from 20 countries, and accepted 15 of them. 8 will be presented orally, and 7 as posters, during a common poster session with the main conference.

A total of 73 students and senior researchers agreed to serve on the program committee, which allowed us to assign 4 reviewers per paper. We would like to thank the reviewers for understanding the spirit of the Student Research Workshop and giving careful and constructive reviews. We hope their comments will be helpful to all the students who submitted their work.

All presenters received financial support from the U.S. National Science Foundation to assist them in their travel to Sydney.

We are very grateful to Rebecca Hwa, our faculty advisor, for her advice, constant support, and obtaining funding. Thanks to Stephen Wan and Chris Callison-Burch, co-chairs of the ACL-05 Student Research Workshop in Ann Arbor, who shared their invaluable experience. Finally, we would like to thank the program chairs of COLING/ACL 2006, Claire Cardie and Pierre Isabelle, the Publications chair Olivia Kwong, as well as the local organization committee, and Priscilla Rasmussen.

The COLING/ACL 2006 Student Research Workshop co-chairs:
Marine Carpuat and Kevin Duh

Organizers

Co-chairs:

Marine Carpuat, Hong Kong University of Science and Technology
Kevin Duh, University of Washington

Faculty advisor:

Rebecca Hwa, University of Pittsburgh

Senior members of the Program Committee:

Eneko Agirre, Basque Country University
Slaven Bilac, Google, Japan
Cem Bozsahin, Middle East Technical Univ
Stephen Clark, Oxford University
Simon Corston-Oliver, Microsoft Research
James Curran, Sydney
Ido Dagan, Bar Ilan University
Pablo Duboue, IBM
Eric Fosler-Lussier, Ohio State
Nizar Habash, Columbia University
Keith Hall, Johns Hopkins University
Mary Hearne, Dublin City University
Philipp Koehn, University of Edinburgh
Emiel Krahmer, Tilburg
Chin Yew Lin, USC/ISI
Bernardo Magnini, ITC irst
Diego Molla Aliod, Macquarie University
Grace Ngai, Hong Kong Polytechnic University
Patrick Pantel, USC/ISI
Cecile Paris, CSIRO
Eric Ringger, Brigham Young University
Graeme Ritchie, Aberdeen
Khalil Sima'an, University of Amsterdam
Michel Simard, NRC Institute for Information Technology
Caroline Sporleder, University of Edinburgh
Nicola Stokes, University of Melbourne
Matthew Stone, Rutgers
Keh-Yih Su, Behavior Design Corporation
Marc Swerts, Tilburg
Joel Tetreault, Pittsburgh
Simone Teufel, Cambridge
Leonoor van der Beek, Groningen
Richard Wicentowski, Swarthmore College
Florian Wolf, FW Consulting
Zhu Zhang, University of Arizona

Student members of the Program Committee:

Michele Banko, University of Washington
Colin Bannard, University of Edinburgh
John Blitzer, University of Pennsylvania
Phil Blunsom, University of Melbourne
Chris Callison-Burch, University of Edinburgh
Xavier Carreras, Technical University of Catalonia (UPC)
Yee Seng Chan, National University of Singapore
Trevor Cohn, University of Melbourne
Tiphaine Dalmas, University of Edinburgh
Hal Daume III, USC/ISI
Etienne Denoual, ATR
Elena Filatova, Columbia University
Erin Fitzgerald, Johns Hopkins University
Michael Fleischman, MIT
Ulrich Germann, Toronto
Teg Grenager, Stanford University
Declan Groves, Dublin City University
Kristy Hollingshead, OGI/Oregon Health and Science University
Matthew Lease, Brown University
Maria Liakata, University of Wales
Gideon Mann, Johns Hopkins University
Ryan McDonald, University of Pennsylvania
Ani Nenkova, Columbia University
Leif Nielsen, King's College
Bo Pang, Cornell
Jean-Philippe Prost, Macquarie University
Sarah Schwarm, University of Washington
Libin Shen, University of Pennsylvania
Yihai Shen, Hong Kong University of Science and Technology
Rion Snow, Stanford University
Charles Sutton, University of Massachussetts, Amherst
Paola Virga, Johns Hopkins University
Stephen Wan, Macquarie University
Ben Wellington, New York University
Theresa Wilson, University of Pittsburgh
Richard Zens, RWTH Aachen
Hao Zhang, University of Rochester
Bing Zhao, CMU

Workshop Program

Monday, 17 July 2006

A Flexible Approach to Natural Language Generation for Disabled Children

Pradipta Biswas
School of Information Technology
Indian Institute of Technology, Kharagpur 721302 INDIA
pbiswas@sit.iitkgp.ernet.in

Abstract

Natural Language Generation (NLG) is a way to automatically realize a correct expression in response to a communicative goal. This technology is mainly explored in the fields of machine translation, report generation, dialog system etc. In this paper we have explored the NLG technique for another novel application-assisting disabled children to take part in conversation. The limited physical ability and mental maturity of our intended users made the NLG approach different from others. We have taken a flexible approach where main emphasis is given on flexibility and usability of the system. The evaluation results show this technique can increase the communication rate of users during a conversation.

1 Introduction

'Natural Language Generation' also known as 'Automated Discourse Generation' or simply 'Text Generation', is a branch of computational linguistics, which deals with automatic generation of text in natural human language by the machine. It can be conceptualized as a process leading from a high level communicative goal to a sequence of communicative acts that accomplish this communicative goal (Rambow et. al., 2001). Based on input representation, any NLG technique can be broadly classified into two paradigms viz. Template based Approach and Plan based approach. The template-based approach does not need large linguistic knowledge resource but it cannot provide the expressiveness or flexibility needed for many real domains (Langkilde and Knight, 1998). In (Deemter et. al., 1999), it has been tried to prove with the example of a system (D2S: Direct to Speech) that both of the approaches are equally powerful and theoretically well founded. The D2S system uses a tree structured template organization that resembles Tag Adjoining Grammar (TAG) structure. The template-based approach that has been taken in the system, enables the basic language generation algorithms application independent and language independent. At the final stage of language generation it checks the compatibility of the sentence structure with the current context and validates the result with Chomsky's binding theory. For this reason it is claimed to be as well founded as any plan-based approach. As another practical example of NLG technique, we can consider the IBM MASTOR system (Liu et. al., 2003). It is used as speech-to-speech translator between English and Mandarin Chinese. The NLG part of this system uses trigram language model for selecting appropriate inflectional form for target language generation.

When NLG (or NLP) technology is applied in assistive technology, the focus is shifted to increase communication rate rather than increasing the efficiency of input representation. As for example, CHAT (Alm, 1992) software is an attempt to develop a predictive conversation model to achieve higher communication rate during conversation. This software predicts different sentences depending on situation and mood of the user. The user is free to change the situation or mood with a few keystrokes. In "Compansion" project (McCoy, 1997), a novel approach was taken to enhance the communication rate. The system takes telegraphic message as input and automatically produces grammatically correct sentences as output based on NLP techniques. The KOMBE Project (Pasero, 1994) tries to enhance the communication rate in a different way. It predicts a sentence or a set of sentence by taking sequence of words from users. The Sanyog project (Sanyog, 2006)(Banerjee, 2005) initiates a dialog with the users to take different portions (eg. Subject, verb, predicate etc.) of a sentence and automatically constructs a grammatically correct sentence based on NLG techniques.

2 The Proposed Approach

The present system is intended to be used by children with severe speech and motor-impairment. It will cater those children who can understand different parts of a sentence (like subject, object, verb etc.) but do not have the competence to construct a grammatically correct sentence by properly arranging words. The intended audience offers both advantages and challenges to our NLG technique. The advantage is we can limit the extent of sentence types that have to be generated. But the challenges overwhelm this advantage. The main challenges identified so far can be summarized as follows.

- Simplicity in interacting with user due to limited mental maturity level of users
- Flexibility in taking input
- Generating sentences with minimum number of keystrokes due to the limited physical ability of the users
- Generating the most appropriate sentence in the first chance since we do not have any scope to provide users a set of sentences and ask them to choose one from the set.

In the next few sections the NLG technique adopted in our system will be discussed in details. Due to limited vocabulary and education level of our intended users, our NLG technique will generate only simple active voice sentences. The challenges are also tried to be addressed in developing the NLG technique.

Generally an NLG system can be divided into three modules viz. Text Planning, MicroPlanning and Realization. In (Callaway and Lester, 1995), the first two modules are squeezed into a planning module and results only two subtasks in an NLG system. Generally in all the approaches of NLG, the process starts with different parts of a sentence and each of these parts can be designated as a template. After getting values for these templates the templates are arranged in a specified order to form an intermediate representation of a sentence. Finally the intermediate representation undergoes through a process viz. Surface realization to form a grammatically correct and fluent sentence. Thus any NLG technique can be broadly divided into two parts

- Templates fill up
- Surface realization

Now each of these two steps for our system will be discussed in details.

2.1 Templates fill up

We defined templates for our system based on thematic roles and Parts of Speech of words. We tagged each sentence of our corpus (the corpus is discussed in section 4.1) and based on this tagged corpus, we have classified the templates in two classes. One class contains the high frequency templates i.e. templates that are contained in most of the sentences. Examples of this class of templates include subject, verb, object etc. The other class contains rest of the templates. Let us consider the first class of templates are designated by set A={a1,a2,a3,a4….} and other class is set B={b1,b2,b3,b4,…………..}.

Our intention is to offer simplicity and flexibility to user during filling up the templates. So each template is associated with an easy to understand phrase like

Subject=> Who
Verb=> Action
Object=> What
Destination=>To Where
Source=>From Where……….etc.

To achieve the flexibility, we show all the templates in set A to user in the first screen (the screenshot is given in fig. 1, however the screen will not look as clumsy as it is shown because some of the options remain hidden by default and appear only on users' request). The user is free to choose any template from set A to start sentence construction and is also free to choose any sequence during filling up values for set A. The system will be a free order natural language generator i.e. user can give input to the system using any order; the system will not impose any particular order on the user (as imposed by the Sanyog Project). Now if the user is to search for all the templates needed for his/her sentence, then both the number of keystrokes and cognitive load on user will increase. So with each template of set A we defined a sequence of templates taking templates from both set A and set B. Let user chooses template ak. Now after filling up template ak, user will be prompted with a sequence of templates like ak1, ak2, ak3, bk1, bk2, bk3, etc. to fill up. Again the actual sequence that will be prompted to user will depend on the input that is already given by user. So the final sequence shown to the user will be a subset of the predefined sequence. Let us clear the concept with an example. Say a user fills up the template <Destination>. Now s/he will be requested to give values for template like <Source>, <Conveyance>, <Time>, <Subject> etc, excluding those which are already filled up. As the example shows, the user needs not to search for all templates as well as s/he needs not to fill up a template more than once. This strategy gives sentence composition with minimum number of keystrokes in most of the cases.

2.2 Surface Realization

It consists of following steps

- ➢ Setting verb form according to the tense given by user
- ➢ Setting Sense
- ➢ Setting Mood
- ➢ Phrase ordering to reflect users intention

Each of these steps is described next.

The verb form will be modified according to the person and number of the subject and the tense choice given by the user.

The sense will decide the type of the sentence i.e. whether it is affirmative, negative, interrogative or optative. For negative sense, appropriate negative word (e.g. No, not, do not) will be inserted before the verb. The relative position of the order of the subject and verb will be altered for optative and interrogative sentences.

The mood choice changes the main verb of the sentence to special verbs like need, must etc. It tries to reflect the mood of the user during sentence composition.

Finally the templates are grouped to constitute different phrases. These phrases are ordered according to the order of the input given by the user. This step is further elaborated in section 3.2.

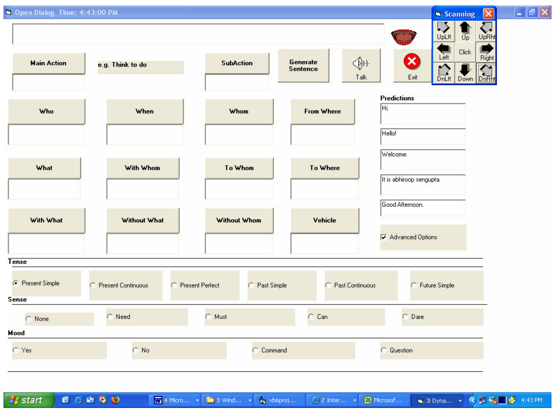
3 A Case Study

In this section a procedural overview of the present system will be described. The automatic language generation mechanism of the present system uses the following steps

Taking Input from Users
The user has to give input to the system using the form shown in fig. 1. As shown in the form the user can select any property (like tense, mood or sense) or template at any order. The user can select tense, mood or sentence type by clicking on appropriate option button. The user can give input for the template by answering to the following questions

- Action
- Who
- Whom
- With Whom
- What
- From Where
- To Where
- Vehicle Used ……etc.

After selecting a thematic role, a second form will come as shown in Fig. 2. From the form shown at Fig 2, the user can select as many words as they want. Even if they want they can type a word (e.g. his /her own name). The punctuations and conjunction will automatically be inserted.



Predictions

Hi.

Hello!

Welcome.

It is abhiroop sengupta.

Good Afternoon.

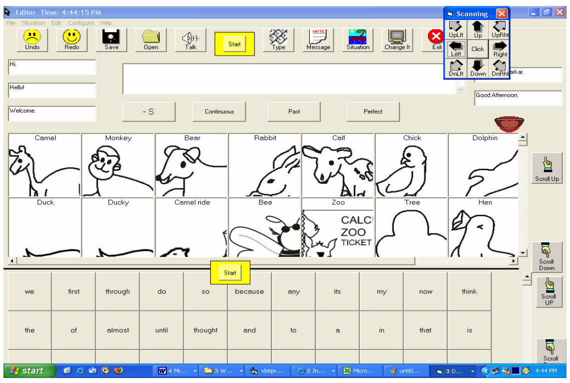
[✓ Advanced Options]

Tense: [✓ Present Simple] Present Continuous Present Perfect Past Simple Past Continuous Future Simple

Sense: None Need Must Can Dare

Mood: Yes No Command Question

Fig. 1: Screenshot of dialog based interface



Hi.

Hello!

Welcome.

arkar.

Good Afternoon.

Fig. 2: Screenshot of word selection interface

Template fill-up
After giving all the input the user asks the system to generate the sentence by clicking on "generate sentence" Button. The system is incorporated with several template organizations and a default

template organization. Examples of some of these template organizations are as follows

- SUBJECT VERB
- SUBJECT VERB INANIMATE OBJECT
- SUBJECT VERB ANIMATE OBJECT
- SUBJECT VERB WITH COAGENT
- SUBJECT VERB INANIMATE OBJECT WITH COAGENT
- SUBJECT VERB INANIMATE OBJECT WITH INSTRUMENT
- SUBJECT VERB SOURCE DESTINATION BY CONVEYANCE
- SUBJECT VERB SOURCE DESTINATION WITH COAGENT

The system select one such template organization based on user input and generates the intermediate sentence representation.

Verb modification according to tense
The intermediate sentence is a simple present tense sentence. According to the user chosen tense, the verb of the intermediate sentence get modified at this step. If no verb is specified, appropriate auxiliary verb will be inserted.

Changing Sentence Type
Up to now the sentence remain as an affirmative sentence. According to the user chosen sense the sentence gets modified in this step. E.g. For question, the verb comes in front, for negative sentence not, do not, did not or does not is inserted appropriately.

Inserting Modal Verbs
Finally the users chosen modal verbs like must, can or need are inserted into the sentence. For some modal verbs (like can or need) the system also changes the form of the verb (like can or could).

3.1 Example of Sentence Generation using Our Approach

Let us consider some example of language generation using our system.

Example 1
Let the user wants to tell, "I am going to school with father"
Step 1: The user inputs will be
Who => I
To Where => school
With Whom => father
Main Action => go
Tense => Present Continuous
Step 2: Template Organization Selection
Based on user input the following template organization will be selected
SUBJECT VERB DESTINATION WITH COAGENT
Step 3: Verb Modification according to tense
Since the selected tense is present continuous and subject is first person singular number, so 'go' will be changed to 'am going'.
Step 4: In this case there is no change of the sentence due to step 4.
Step 5: There is no change of the sentence due to step 5.
So the final output will be "I am going to school with father". It is same as the user intended sentence.

Example 2
Let the user wants to tell, "You must eat it"
Step 1: The user inputs will be
Who => You
Main Action => eat
What => it
Mood => must
Tense => Present Simple
Step 2: Template Organization Selection
Based on user input the following template organization will be selected
SUBJECT VERB INANIMATE OBJECT
Step 3: Verb Modification according to tense
Since the tense is present simple so there will be no change in verb.
Step 4: In this case there is no change of the sentence due to step 4.
Step 5: The modal verb will be inserted before the verb
So the final output will be "You must eat it"

Example 3
Let the user wants to tell, "How are you"
Step 1: The user inputs will be
Who => You
Sense => Question
Wh-word => How
Tense => Present Simple
Step 2: Template Organization Selection
There is no appropriate template for this input. Hence the default template organization will be chosen.
Step 3: Verb Modification according to tense

Since no action is specified, the auxiliary verb will be selected as the main verb. Here the subject is second person and tense is present simple, so the verb selected is 'are'.

Step 4: Since the selected sentence type is 'Question', so the verb will come in front of the sentence. Again, since a Wh-word has been selected, it will come in front of the verb. A question mark will automatically be appended at the end of the sentence.

Step 5: There is no change of the sentence due to step 5.

So the final output will be "How are you?"

3.2 Phase ordering to reflect users' intention

An important part of any NLG system is pragmatics that can be defined as the reference to the interlocutors and context in communication (Hovy, 1990). In (Hovy, 1990), a system viz. PAULINE has been described that is capable of generating different texts for the same communicative goals based on pragmatics. In PAULINE, the pragmatics has been represented by rhetorical goals. The rhetorical goals defined several situations that dictate all the phases like topic collection, topic organization and realization. Inspired from the example of PAULINE the present system has also tried to reflect users' intention during sentence realization. Here the problem is the limited amount of input for making any judicious judgment. The input to the system is only a sequence of words with correspondence to a series of questions. A common finding is that we uttered the most important concept in a sentence earlier than other parts of the sentence. So we have tried to get the users' intention from the order of input given by user based on the belief that the user will fill up the slots in order of their importance according to his/her mood at that time. We have associated a counter with each template. The counter value is taken from a global clock that is updated with each word selection by the user. Each sentence is divided into several phrases before realization. Now each phrase constitute of several templates. For example let S be a sentence. Now S can be divided into phrases like P1, P2, P3….. Again each phrase Pi can be divided into several templates like T1, T2, T3….Based on the counter value of each template, we have calculated the rank of each phrase as the minimum counter value of its constituent templates i.e.

Rank(Pi)=Minimum(Counter(Tj)) for all j in Pi

Now before sentence realization the phrases are ordered according to their rank. Each of these phrase orders produces a separate sentence. As for example let the communication goal is *'I go to school from home with my father'*. If the input sequence is (my father -> I -> go -> school -> home), the generated sentence will be *'With my father I go from home to School'*. Again if the input sequence is (school -> home -> I -> go -> my father), then the generated sentence will be *'From home to school I go with my father.'*

Thus for the same communicative goal, the system produces different sentences based on order of input given by user.

4 Evaluation

The main goal of our system is to develop a communication aid for disabled children. So the performance metrics concentrated on measuring the communication rate that has little importance from NLG point of view. To evaluate our system from NLG point of view we emphasize on the expressiveness and ease of use of the system. The expressiveness is measured by the percentage of sentences that was intended by user and also successfully generated by our system. The ease of use is measured by the average number of inputs needed to generate each sentence.

4.1 Measuring Expressiveness

To know the type of sentences used by our intended users during conversation, first we analyzed the communication boards used by disabled children. Then we took part in some actual conversations with some spastic children in a Cerebral Palsy institute. Finally we interviewed their teachers and communication partners. Based on our research, we developed a list of around 1000 sentences that covers all types of sentences used during conversation. This list is used as a corpus in both development and evaluation stage of our system. During development the corpus is used to get the necessary templates and for classification of templates (refer sec. 2.1). After development, we tested the scope of our system by generating some sentences that were exactly not in our corpus, but occurred in some sample conversations of the intended users. In 96% cases, the system is successful to generate the intended sentence. After analyzing the rest 4% of sentence, we have identified following problems at the current implementation stage.

- ➢ The system cannot handle gerunds as object to preposition. (e.g. He ruins his eyes *by reading* small letters).
- ➢ The system is yet not capable to generate correct sentence with an introductory ‘It’. (e.g. It is summer). In these situations the sentence is correctly generated when ‘It’ is given as an agent, which is not intended.

4.2 Measuring ease of use

To calculate the performance of the system, we measured the number of inputs given by user for generating sentence. The input consists of words, tense choice, mood option and sense choice given by user. Next we plot the number of inputs w.r.t. the number of words for each sentence. Fig. 3 shows the plot. It can be observed from the plot that as the number of words increases (i.e. for longer sentences), the ratio of number of inputs to number of words decreases. So effort from users’ side will not vary remarkably with sentence length. The overall communication rate is found to be 5.52 words/min (27.44 characters/min) that is better than (Stephanidis, 2003). Additionally it is also observed that the communication rate is increasing with longer conversations.

5 Conclusion

The present paper discusses a flexible approach for natural language generation for disabled children. A user can start a sentence generation from any part of a sentence. The inherent sentence plan will guide him to realize a grammatically correct sentence with minimum number of keystrokes. The present system respects the pragmatics of a conversation by reordering different parts of a sentence following users’ intention. The system is evaluated both from expressiveness and performance point of views. Initial evaluation results show this approach can increase the communication rate of intended users during conversation.

Acknowledgement

The author is grateful to Media Lab Asia Laboratory of IIT Kharagpur and Indian Institute of Cerebral Palsy, Kolkata for exchanging ideas and providing resources for the present work.

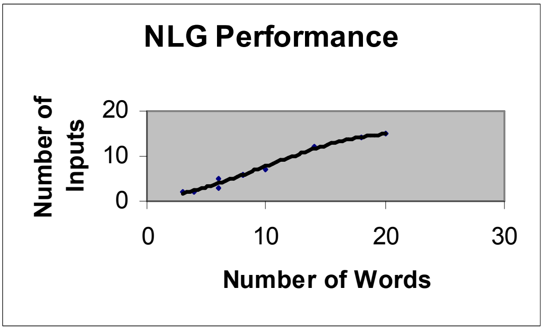


Fig. 3: Line graph for performance measurement of the system

References

Alm N., Arnott J. L., Newell A. F. 1992, Prediction and Conversational Momentum in an Augmentative Communication System, Communications of the ACM, vol. 55, No. 5, May 1992

Banerjee A. 2005, A Natural Language Generation Framework for an Interlingua-based Machine Translation System, MS Thesis, IIT Kharagpur

Callaway Charles B., Lester James C. 1995, Robust Natural Language Generation from Large-Scale Knowledge Bases, Proceedings of the Fourth Bar-Ilan Symposium on Foundations

Hovy E. H. 1990, Pragmatics and Natural Language Generation, Artificial Inteligence 43(1990): 153-197

Liu Fu-Hua,Liang Gao Gu,Yuqing, Picheny Michael 2003, Use of Statistical N_Gram Models in Natural Language Generation for Machine translation, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Vol 1 : 636 639

Langkilde Irene, Knight Kevin 1998, Generation that Exploits Corpus-Based Statistical Knowledge, Annual meeting-association for computational linguistics: 704-710

Deemter Kees van et. al. 1999, Plan-Based vs. template-based NLG: a false opposition?, Becker and Busemann (1999)

McCoy K. 1997 , “Simple NLP Techniques for Expanding Telegraphic Sentences” Natural Language Processing for Communication Aids,1997

Rambow Owen, Bangalore Srinivas, Walker Marilyn 2001,Natural Language Generation in Dialog System, Proceedings of the first international conference on Human language technology research HLT '01

Pasero Robert, Nathalie Richardet and Paul Sabatier; “Guided Sentences Composition for Disabled People”; Proceedings of the fourth conference on Applied natural language processing October 1994

Project SANYOG Available at: http://www.mla.iitkgp.ernet.in/projects/sanyog.htm

Stephanidis, C. et. al., “Designing Human Computer Interfaces for Quadriplegic People”, ACM Transactions on Computer-Human Interaction, pp 87-118, Vol. 10, No. 2, June 2003

Unsupervised Part-of-Speech Tagging Employing Efficient Graph Clustering

Chris Biemann
University of Leipzig, NLP Department
Augustusplatz 10/11, 04109 Leipzig, Germany
biem@informatik.uni-leipzig.de

Abstract

An unsupervised part-of-speech (POS) tagging system that relies on graph clustering methods is described. Unlike in current state-of-the-art approaches, the kind and number of different tags is generated by the method itself. We compute and merge two partitionings of word graphs: one based on context similarity of high frequency words, another on log-likelihood statistics for words of lower frequencies. Using the resulting word clusters as a lexicon, a Viterbi POS tagger is trained, which is refined by a morphological component. The approach is evaluated on three different languages by measuring agreement with existing taggers.

1 Introduction

1.1 Motivation

Assigning syntactic categories to words is an important pre-processing step for most NLP applications.

Essentially, two things are needed to construct a tagger: a lexicon that contains tags for words and a mechanism to assign tags to running words in a text. There are words whose tags depend on their use. Further, we also need to be able to tag previously unseen words. Lexical resources have to offer the possible tags, and our mechanism has to choose the appropriate tag based on the context.

Given a sufficient amount of manually tagged text, several approaches have demonstrated the ability to learn the instance of a tagging mechanism from manually labelled data and apply it successfully to unseen data. Those high-quality resources are typically unavailable for many languages and their creation is labour-intensive. We will describe an alternative needing much less human intervention.

In this work, steps are undertaken to derive a lexicon of syntactic categories from unstructured text without prior linguistic knowledge. We employ two different techniques, one for high- and medium frequency terms, one for medium- and low frequency terms. The categories will be used for the tagging of the same text where the categories were derived from. In this way, domain- or language-specific categories are automatically discovered.

1.2 Existing Approaches

There are a number of approaches to derive syntactic categories. All of them employ a syntactic version of Harris' distributional hypothesis: Words of similar parts of speech can be observed in the same syntactic contexts. Contexts in that sense are often restricted to the most frequent words. The words used to describe syntactic contexts will be called *feature words* in the remainder. *Target words*, as opposed to this, are the words that are to be grouped into syntactic clusters.

The general methodology (Finch and Chater, 1992; Schütze, 1995; inter al.) for inducing word class information can be outlined as follows:

1. Collect global context vectors for target words by counting how often feature words appear in neighbouring positions.
2. Apply a clustering algorithm on these vectors to obtain word classes

Throughout, feature words are the 150-250 words with the highest frequency. Contexts are the feature words appearing in the immediate neighbourhood of a word. The word's global context is the sum of all its contexts.

For clustering, a similarity measure has to be defined and a clustering algorithm has to be chosen. Finch and Chater (1992) use the Spearman Rank Correlation Coefficient and a hierarchical clustering, Schütze (1995) uses the cosine between vector angles and Buckshot clustering.

An extension to this generic scheme is presented in (Clark, 2003), where morphological information is used for determining the word class of rare words. Freitag (2004) does not sum up the contexts of each word in a context vector, but the most frequent instances of four-word windows are used in a co-clustering algorithm.

Regarding syntactic ambiguity, most approaches do not deal with this issue while clustering, but try to resolve ambiguities at the later tagging stage.

A severe problem with most clustering algorithms is that they are parameterised by the number of clusters. As there are as many different word class schemes as tag sets, and the exact amount of word classes is not agreed upon intra- and interlingually, inputting the number of desired clusters beforehand is clearly a drawback. In that way, the clustering algorithm is forced to split coherent clusters or to join incompatible sub-clusters. In contrast, unsupervised part-of-speech induction means the induction of the tag set, which implies finding the number of classes in an unguided way.

1.3 Outline

This work constructs an unsupervised POS tagger from scratch. Input to our system is a considerable amount of unlabeled, monolingual text bar any POS information. In a first stage, we employ a clustering algorithm on distributional similarity, which groups a subset of the most frequent 10,000 words of a corpus into several hundred clusters (partitioning 1). Second, we use similarity scores on neighbouring co-occurrence profiles to obtain again several hundred clusters of medium- and low frequency words (partitioning 2). The combination of both partitionings yields a set of word forms belonging to the same derived syntactic category. To gain on text coverage, we add ambiguous high-frequency words that were discarded for partitioning 1 to the lexicon. Finally, we train a Viterbi tagger with this lexicon and augment it with an affix classifier for unknown words.

The resulting taggers are evaluated against outputs of supervised taggers for various languages.

2 Method

The method employed here follows the coarse methodology as described in the introduction, but differs from other works in several respects. Although we use 4-word context windows and the top frequency words as features (as in Schütze 1995), we transform the cosine similarity values between the vectors of our target words into a graph representation. Additionally, we provide a methdology to identify and incorporate POS-ambiguous words as well as low-frequency words into the lexicon.

2.1 The Graph-Based View

Let us consider a weighted, undirected graph *G(V,E)* ($v \in V$ vertices, $(v_i,v_j,w_{ij}) \in E$ edges with weights w_{ij}). Vertices represent entities (here: words); the weight of an edge between two vertices indicates their similarity.

As the data here is collected in feature vectors, the question arises why it should be transformed into a graph representation. The reason is, that graph-clustering algorithms such as e.g. (van Dongen, 2000; Biemann 2006), find the number of clusters automatically[1]. Further, outliers are handled naturally in that framework, as they are represented as singleton nodes (without edges) and can be excluded from the clustering. A threshold *s* on similarity serves as a parameter to influence the number of non-singleton nodes in the resulting graph.

For assigning classes, we use the Chinese Whispers (CW) graph-clustering algorithm, which has been proven useful in NLP applications as described in (Biemann 2006). It is time-linear with respect to the number of edges, making its application viable even for graphs with several million nodes and edges. Further, CW is parameter-free, operates locally and results in a partitioning of the graph, excluding singletons (i.e. nodes without edges).

2.2 Obtaining the lexicon

Partitioning 1: High and medium frequency words

Four steps are executed in order to obtain partitioning 1:

1. Determine 200 feature and 10.000 target words from frequency counts
2. construct graph from context statistics
3. Apply CW on graph.
4. Add the feature words not present in the partitioning as one-member clusters.

The graph construction in step 2 is conducted by adding an edge between two words a and b

[1] This is not an exclusive characteristic for graph clustering algorithms. However, the graph model deals with that naturally while other models usually build some meta-mechanism on top for determining the optimal number of clusters.

with weight w=1/(1-cos(a,b)), if w exceeds a similarity threshold *s*. The latter influences the number of words that actually end up in the graph and get clustered. It might be desired to cluster fewer words with higher confidence as opposed to running in the danger of joining two unrelated clusters because of too many ambiguous words that connect them.

After step 3, we already have a partition of a subset of our target words. The distinctions are normally more fine-grained than existing tag sets.

As feature words form the bulk of tokens in corpora, it is clearly desired to make sure that they appear in the final partitioning, although they might form word classes of their own[2]. This is done in step 4. We argue that assigning separate word classes for high frequency words is a more robust choice then trying to disambiguate them while tagging.

Lexicon size for partitioning 1 is limited by the computational complexity of step 2, which is time-quadratic in the number of target words. For adding words with lower frequencies, we pursue another strategy.

Partitioning 2: Medium and low frequency words

As noted in (Dunning, 1993), log-likelihood statistics are able to capture word bi-gram regularities. Given a word, its neighbouring co-occurrences as ranked by the log-likelihood reflect the typical immediate contexts of the word. Regarding the highest ranked neighbours as the profile of the word, it is possible to assign similarity scores between two words A and B according to how many neighbours they share, i.e. to what extent the profiles of A and B overlap. This directly induces a graph, which can be again clustered by CW.

This procedure is parametrised by a log-likelihood threshold and the minimum number of left and right neighbours A and B share in order to draw an edge between them in the resulting graph. For experiments, we chose a minimum log-likelihood of 3.84 (corresponding to statistical dependence on 5% level), and at least four shared neighbours of A and B on each side.

Only words with a frequency rank higher than 2,000 are taken into account. Again, we obtain several hundred clusters, mostly of open word classes. For computing partitioning 2, an efficient algorithm like CW is crucial: the graphs as used for the experiments consisted of 52,857/691,241 (English), 85,827/702,349 (Finnish) and 137,951/1,493,571 (German) nodes/edges.

[2] This might even be desired, e.g. for English *not.*

The procedure to construct the graphs is faster than the method used for partitioning 1, as only words that share at least one neighbour have to be compared and therefore can handle more words with reasonable computing time.

Combination of partitionings 1 and 2

Now, we have two partitionings of two different, yet overlapping frequency bands. A large portion of these 8,000 words in the overlapping region is present in both partitionings. Again, we construct a graph, containing the clusters of both partitionings as nodes; weights of edges are the number of common elements, if at least two elements are shared. And again, CW is used to cluster this graph of clusters. This results in fewer clusters than before for the following reason: While the granularities of partitionings 1 and 2 are both high, they capture different aspects as they are obtained from different sources. Nodes of large clusters (which usually consist of open word classes) have many edges to the other partitioning's nodes, which in turn connect to yet other clusters of the same word class. Eventually, these clusters can be grouped into one.

Clusters that are not included in the graph of clusters are treated differently, depending on their origin: clusters of partition 1 are added to the result, as they are believed to contain important closed word class groups. Dropouts from partitioning 2 are left out, as they mostly consist of small, yet semantically motivated word sets. Combining both partitionings in this way, we arrive at about 200-500 clusters that will be further used as a lexicon for tagging.

Lexicon construction

A lexicon is constructed from the merged partitionings, which contains one possible tag (the cluster ID) per word. To increase text coverage, it is possible to include those words that dropped out in the distributional step for partitioning 1 into the lexicon. It is assumed that these words dropped out because of ambiguity. From a graph with a lower similarity threshold *s* (here: such that the graph contained 9,500 target words), we obtain the neighbourhoods of these words one at a time. The tags of those neighbours – if known – provide a distribution of possible tags for these words.

2.3 Constructing the tagger

Unlike in supervised scenarios, our task is not to train a tagger model from a small corpus of hand-tagged data, but from our clusters of derived syntactic categories and a considerably large, yet unlabeled corpus.

Basic Trigram Model

We decided to use a simple trigram model without re-estimation techniques. Adopting a standard POS-tagging framework, we maximize the probability of the joint occurrence of tokens (t_i) and categories (c_i) for a sequence of length *n*:

$$P(T,C) = \prod_{i=1}^{n} P(c_i \mid c_{i-1}, c_{i-2}) P(c_i \mid t_i).$$

The transition probability $P(c_i|c_{i-1},c_{i-2})$ is estimated from word trigrams in the corpus whose elements are all present in our lexicon.

The last term of the product, namely $P(c_i|t_i)$, is dependent on the lexicon[3]. If the lexicon does not contain (t_i), then (c_i) only depends on neighbouring categories. Words like these are called out-of-vocabulary (OOV) words.

Morphological Extension

Morphologically motivated add-ons are used e.g. in (Clark, 2003) and (Freitag 2004) to guess a more appropriate category distribution based on a word's suffix or its capitalization for OOV words. Here, we examine the effects of Compact Patricia Trie classifiers (CPT) trained on prefixes and suffixes. We use the implementation of (Witschel and Biemann, 2005). For OOV words, the category-wise product of both classifier's distributions serve as probabilities $P(c_i|t_i)$: Let *w=ab=cd* be a word, *a* be the longest common prefix of *w* that can be found in all lexicon words, and *d* be the longest common suffix of *w* that can be found in all lexicon words. Then

$$P(c_i \mid w) = \frac{\left|\{u \mid u = ax \wedge \text{class}(u) = c_i\}\right|}{\left|\{u \mid u = ax\}\right|} \bullet \frac{\left|\{v \mid v = yd \wedge \text{class}(v) = c_i\}\right|}{\left|\{v \mid v = yd\}\right|}$$

CPTs do not only smoothly serve as a substitute lexicon component, they also realize capitalization, camel case and suffix endings naturally.

3 Evaluation methodology

We adopt the methodology of (Freitag 2004) and measure *cluster-conditional tag perplexity* PP as the average amount of uncertainty to predict the tags of a POS-tagged corpus, given the tagging with classes from the unsupervised method. Let

$$I_X = -\sum_x P(x) \ln P(x)$$

be the entropy of a random variable X and

$$M_{XY} = \sum_{xy} P(x,y) \ln \frac{P(x,y)}{P(x)P(y)}$$

be the mutual information between two random variables X and Y. Then the cluster-conditional tag perplexity for a gold-standard tagging T and a tagging resulting from clusters C is computed as

$$PP = \exp(I_{T|C}) = \exp(I_T - M_{TC}).$$

Minimum PP is 1.0, connoting a perfect congruence on gold standard tags.

In the experiment section we report PP on lexicon words and OOV words separately. The objective is to minimize the total PP.

4 Experiments

4.1 Corpora

For this study, we chose three corpora: the British National Corpus (BNC) for English, a 10 Million sentences newspaper corpus from Projekt Deutscher Wortschatz[4] for German, and 3 million sentences from a Finnish web corpus (from the same source). Table 1 summarizes some characteristics.

lang.	sent.	tok.	tagger	nr. tags	200 cov.	10K cov.
en	6M	100M	BNC[5]	84	55%	90%
fi	3M	43M	Connexor[6]	31	30%	60%
ger	10M	177M	(Schmid,1994)	54	49%	78%

Table 1: Characteristics of corpora: number of sentences, tokens, tagger and tagset size, corpus coverage of top 200 and 10,000 words.

Since a high coverage is reached with few words in English, a strategy that assigns only the most frequent words to sensible clusters will take us very far here. In the Finnish case, we can expect a high OOV rate, hampering performance

[3] Although (Charniak et al. 1993) report that using $P(t_i|c_i)$ instead leads to superior results in the supervised setting, we use the 'direct' lexicon probability. Note that our training material size is considerably larger than hand-labelled POS corpora.

[4] See http://corpora.informatik.uni-leipzig.de.
[5] Semi-automatic tags as provided by BNC.
[6] Thanks goes to www.connexor.com for an academic license; the tags do not include interpunctuation marks, which are treated seperately.

of strategies that cannot cope well with low frequency or unseen words.

4.2 Baselines

To put our results in perspective, we computed the following baselines on random samples of the same 1000 randomly chosen sentences that we used for evaluation:

- *1*: the trivial top clustering: all words are in the same cluster
- *200*: The most frequent 199 words form clusters of their own; all the rest is put into one cluster.
- *400*: same as 200, but with 399 most frequent words

Table 2 summarizes the baselines. We give PP figures as well as tag-conditional cluster perplexity PP_G (uncertainty to predict the clustering from the gold standard tags, inverse direction of PP):

lang	English			Finnish			German		
base	*1*	*200*	*400*	*1*	*200*	*400*	*1*	*200*	*400*
PP	29	3.6	3.1	20	6.1	5.3	19	3.4	2.9
PP_G	1.0	2.6	3.5	1.0	2.0	2.5	1.0	2.5	3.1

Table 2: Baselines for various tag set sizes

4.3 Results

We measured the quality of the resulting taggers for combinations of several substeps:

- **O:** Partitioning 1
- **M:** the CPT morphology extension
- **T:** merging partitioning 1 and 2
- **A:** adding ambiguous words to the lexicon

Figure 2 illustrates the influence of the similarity threshold *s* for O, OM and OMA for German – the other languages showed similar results. Varying *s* influences coverage on the 10,000 target words. When clustering very few words, tagging performance on these words reaches a PP as low as 1.25 but the high OOV rate impairs the total performance. Clustering too many words results in deterioration of results - most words end up in one big partition. In the medium ranges, higher coverage and lower known PP compensate each other, optimal total PPs were observed at target coverages 4,000-8,000. Adding ambiguous words results in a worse performance on lexicon words, yet improves overall performance, especially for high thresholds.

For all further experiments we fixed the threshold in a way that partitioning 1 consisted of 5,000 words, so only half of the top 10,000 words are considered unambiguous. At this value, we found the best performance averaged over all corpora.

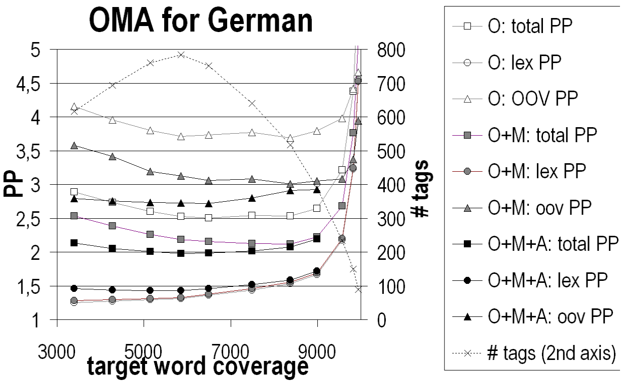


Fig 2. Influence of threshold s on tagger performance: cluster-conditional tag perplexity PP as a function of target word coverage.

lang		O	OM	OMA	TM	TMA
EN	total	2.66	2.43	2.08	2.27	2.05
	lex	1.25		1.51	1.58	1.83
	oov	6.74	6.70	5.82	9.89	7.64
	oov%	28.07		14.25	14.98	4.62
	tags	619			345	
FI	total	4.91	3.96	3.79	3.36	3.22
	lex	1.60		2.04	1.99	2.29
	oov	8.58	7.90	7.05	7.54	6.94
	oov%	47.52		36.31	32.01	23.80
	tags	625			466	
GER	total	2.53	2.18	1.98	1.84	1.79
	lex	1.32		1.43	1.51	1.57
	oov	3.71	3.12	2.73	2.97	2.57
	oov%	31.34		23.60	19.12	13.80
	tags	781			440	

Table 3: results for English, Finnish, German. oov% is the fraction of non-lexicon words.

Overall results are presented in table 3. The combined strategy TMA reaches the lowest PP for all languages. The morphology extension (M) always improves the OOV scores. Adding ambiguous words (A) hurts the lexicon performance, but largely reduces the OOV rate, which in turn leads to better overall performance. Combining both partitionings (T) does not always decrease the total PP a lot, but lowers the number of tags significantly. Finnish figures are generally worse than for the other languages, akin to higher baselines.

The high OOV perplexities for English in experiment TM and TMA can be explained as follows: The smaller the OOV rate gets, the more likely it is that the corresponding words were also OOV in the gold standard tagger. A remedy would be to evaluate on hand-tagged data. Differences between languages are most obvious when comparing OMA and TM: whereas for English it pays off much more to add ambiguous words than to merge the two partitionings, it is the other way around in the German and Finnish experiments.

To wrap up: all steps undertaken improve the performance, yet their influence's strength varies. As a flavour of our system's output, consider the example in table 4 that has been tagged by our English TMA model: as in the introductory example, "saw" is disambiguated correctly.

Word	cluster ID	cluster members (size)
I	166	I (1)
saw	2	*past tense verbs* (3818)
the	73	a, an, the (3)
man	1	*nouns* (17418)
with	13	*prepositions* (143)
a	73	a, an, the (3)
saw	1	*nouns* (17418)
.	116	. ! ? (3)

Table 4: Tagging example

We compare our results to (Freitag, 2004), as most other works use different evaluation techniques that are only indirectly measuring what we try to optimize here. Unfortunately, (Freitag 2004) does not provide a total PP score for his 200 tags. He experiments with an hand-tagged, clean English corpus we did not have access to (the Penn Treebank). Freitag reports a PP for known words of 1.57 for the top 5,000 words (91% corpus coverage, baseline 1 at 23.6), a PP for unknown words without morphological extension of 4.8. Using morphological features the unknown PP score is lowered to 4.0. When augmenting the lexicon with low frequency words via their distributional characteristics, a PP as low as 2.9 is obtained for the remaining 9% of tokens. His methodology, however, does not allow for class ambiguity in the lexicon, the low number of OOV words is handled by a Hidden Markov Model.

5 Conclusion and further work

We presented a graph-based approach to unsupervised POS tagging. To our knowledge, this is the first attempt to leave the decision on tag granularity to the tagger. We supported the claim of language-independence by validating the output of our system against supervised systems in three languages.

The system is not very sensitive to parameter changes: the number of feature words, the frequency cutoffs, the log-likelihood threshold and all other parameters did not change overall performance considerably when altered in reasonable limits. In this way it was possbile to arrive at a one-size-fits-all configuration that allows the parameter-free unsupervised tagging of large corpora.

To really judge the benefit of an unsupervised tagging system, it should be evaluated in an application-based way. Ideally, the application should tell us the granularity of our tagger: e.g. semantic class learners could greatly benefit from the high-granular word sets arising in both of our partitionings, which we endeavoured to lump into a coarser tagset here.

References

C. Biemann. 2006. *Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems*. Proceedings of the HLT-NAACL-06 Workshop on Textgraphs-06, New York, USA

E. Charniak, C. Hendrickson, N. Jacobson and M. Perkowitz. 1993. *Equations for part-of-speech tagging*. In Proceedings of the 11th National Conference on AI, pp. 784-789, Menlo Park

A. Clark. 2003. *Combining Distributional and Morphological Information for Part of Speech Induction*, Proceedings of EACL-03

T. Dunning. 1993. *Accurate Methods for the Statistics of Surprise and Coincidence*, Computational Linguistics 19(1), pp. 61-74

S. Finch and N. Chater. 1992. *Bootstrapping Syntactic Categories Using Statistical Methods*. In Proc. 1st SHOE Workshop. Tilburg, The Netherlands

D. Freitag. 2004. *Toward unsupervised whole-corpus tagging*. Proc. of COLING-04, Geneva, 357-363.

H. Schmid. 1994. *Probabilistic Part-of-Speech Tagging Using Decision Trees*. In: Proceedings of the International Conference on New Methods in Language Processing, Manchester, UK, pp. 44-49

H. Schütze. 1995. *Distributional part-of-speech tagging*. In EACL 7, pages 141–148

S. van Dongen. 2000. *A cluster algorithm for graphs*. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam.

F. Witschel, and C. Biemann. 2005. *Rigorous dimensionality reduction through linguistically motivated feature selection for text categorisation*. Proc. of NODALIDA 2005, Joensuu, Finland

Sub-sentential Alignment Using Substring Co-Occurrence Counts

Fabien Cromieres
GETA-CLIPS-IMAG
BP53 38041 Grenoble Cedex 9
France
fabien.cromieres@gmail.com

Abstract

In this paper, we will present an efficient method to compute the co-occurrence counts of any pair of substring in a parallel corpus, and an algorithm that make use of these counts to create sub-sentential alignments on such a corpus. This algorithm has the advantage of being as general as possible regarding the segmentation of text.

1 Introduction

An interesting and important problem in the Statistical Machine Translation (SMT) domain is the creation of sub-sentential alignment in a parallel corpus (a bilingual corpus already aligned at the sentence level). These alignments can later be used to, for example, train SMT systems or extract bilingual lexicons.

Many algorithms have already been proposed for sub-sentential alignment. Some of them focus on word-to-word alignment ((Brown,97) or (Melamed,97)). Others allow the generation of phrase-level alignments, such as (Och et al., 1999), (Marcu and Wong, 2002) or (Zhang, Vogel, Waibel, 2003). However, with the exception of Marcu and Wong, these phrase-level alignment algorithms still place their analyses at the word level; whether by first creating a word-to-word alignment or by computing correlation coefficients between pairs of individual words.

This is, in our opinion, a limitation of these algorithms; mainly because it makes them rely heavily on our capacity to segment a sentence in words. And defining what a word is is not as easy as it might seem. In peculiar, in many Asians writings systems (Japanese, Chinese or Thai, for example), there is not a special symbol to delimit words (such as the blank in most non-Asian writing systems). Current systems usually work around this problem by using a segmentation tool to pre-process the data. There are however two major disadvantages:

- These tools usually need a lot of linguistic knowledge, such as lexical dictionaries and hand-crafted segmentation rules. So using them somehow reduces the "purity" and universality of the statistical approach.
- These tools are not perfect. They tend to be very dependent on the domain of the text they are used with. Besides, they cannot take advantage of the fact that there exist a translation of the sentence in another language.

(Xu, Zens and Ney,2004) have overcome part of these objections by using multiple segmentations of a Chinese sentence and letting a SMT system choose the best one, as well as creating a segmentation lexicon dictionary by considering every Chinese character to be a word in itself and then creating a phrase alignment. However, it is probable that this technique would meet much more difficulties with Thai, for example (whose characters, unlike Chinese, bear no specific sense) or even Japanese (which use both ideograms and phonetic characters).

Besides, even for more "computer-friendly" languages, relying too much on typographic words may not be the best way to create an alignment. For example, the translation of a set phrase may contain no word that is a translation of the individual words of this set phrase. And one could consider languages such as German, which tend to merge words that are in relation in a single typographic word. For such languages, it could be a good thing to be able to create alignment at an even more basic level than the typographic words.

These thoughts are the main motivations for the development of the alignment algorithm we will expose in this paper. Its main advantage is that it can be applied whatever is the smallest unit of text we want to consider: typographic word or single character. And even when working at the character level, it can use larger sequence of characters to create correct alignments. The problem of the segmentation and of the alignment will be resolved simultaneously: a sentence and its translation will mutually induce a segmentation on one another. Another advantage of this algorithm is that it is purely statistical: it will not require any information other than the parallel corpus we want to align.

It should be noted here that the phrase-level joint-probability model presented in (Marcu and Wong) can pretend to have the same qualities. However, it was only applied to word-segmented texts by its authors. Making use of the EM training, it is also much more complex than our approach.

Before describing our algorithm, we will explain in detail a method for extracting the co-occurrence counts of any substring in a parallel corpus. Such co-occurrence counts are important to our method, but difficult to compute or store in the case of big corpora.

2 Co-Occurrence counting algorithm

2.1 Notation and definitions

In the subsequent parts of this paper, a substring will denote indifferently a sequence of characters or a sequence of words (or actually a sequence of any typographic unit we might want to consider). The terms "elements" will be used instead of "word" or "characters" to denote the fundamental typographic unit we chose for a given language.

In general, the number of co-occurrences of two substrings S_1 and S_2 in a parallel corpus is the number of times they have appeared on the opposite sides of a bi-sentence in this corpus. It will be noted $N(S_1,S_2)$. In the cases where S_1 and S_2 appears several times in a single bi-sentence (n_1 and n_2 times respectively), we might count 1, n_1*n_2 or $min(n_1,n_2)$ co-occurrences. We will also note $N(S_1)$ the number of occurrences of S_1 in the corpus.

2.2 The Storage Problem

Counting word co-occurrences over a parallel corpus and storing them in a data structure such as a Hash table is a trivial task. But storing the co-occurrences counts of every pair of substring presents much more technical difficulties. Basically, the problem is that the number of values to be stored is much greater when we consider substrings. For two sentences with N_1 and N_2 words respectively, there are N_1*N_2 words that co-occur; but the number of substrings that co-occur is roughly proportional to (N_1*N_2)^2. Of course, most substrings in a pair of sentences are not unique in the corpus, which reduces the number of values to be stored. Still, in most cases, it remains impractical. For example, the Japanese-English BTEC corpus has more than 11 million unique English (word-) substrings and more than 8 million unique Japanese (character-) substrings. So there are potentially 88,000 billion co-occurrence values to be stored. Again, most of these substrings do not co-occur in the corpus, so that non-zero co-occurrences values are only a fraction of this figure. However, a rough estimation we performed showed that there still would be close to a billion values to store.

With a bigger corpus such as the European Parliament Corpus (more than 600,000 sentences per languages) we have more than 698 millions unique English (word-) substrings and 875 millions unique French (word-) substrings. And things get much worse if we want to try to work with characters substrings.

To handle this problem, we decided not to try and store the co-occurrences count beforehand, but rather to compute them "on-the-fly", when they are needed. For that we will need a way to compute co-occurrences very efficiently. We will show how to do it with the data structure known as Suffix Array.

2.3 Suffix Arrays

Suffix Arrays are a data structure allowing for (among other things) the efficient computation of the number of occurrences of any substring within a text. They have been introduced by Mamber and Myers (1993) in a bioinformatics context. (Callison-Burch, Bannard and Scroeder, 2005) used them (in a way similar to us) to compute and store phrase translation probabilities over very large corpora.

Basically, a Suffix Array is a very simple data structure: it is the sorted list of all the suffixes of a text. A suffix is a substring going from one starting position in the text to its end. So a text of T elements has T suffixes.

An important point to understand is that we won't have to store the actual suffixes in memory. We can describe any suffix by its starting position in the text. Hence, every suffix occupies a constant space in memory. Actually, a common implementation is to represent a suffix by a memory pointer on the full text. So, on a machine with 32-bit pointers, the Suffix Array of a text of T elements occupy $4*T$ bytes. The time complexity of the Suffix Array construction is $O(T*log(T))$ if we build the array of the suffixes and then sort it.

We will now describe the property of the Suffix Array that interest us. Let S be a substring. Let pf be the position (in the Suffix Array) of the first suffix beginning with substring S and pl be the position of the last such suffix. Then every suffix in the Array between positions pf and pl corresponds to an occurrence of S. And every occurrence of S in the text corresponds to a suffix between pf and pl.

pf and pl can be retrieved in $O(|S|*log\ T)$ with a dichotomy search. Beside, $N(S)=pl-pf+1$; so we can compute $N(S)$ in $O(|S|*log\ T)$. We will now see how to compute $N(S_1,S_2)$ for two substrings S_1 and S_2 in a parallel corpus.

2.4 Computing Co-Occurrences using Suffix Array

A Suffix Array can be created not only from one text, but also from a sequence of texts. In the present case, we will consider the sequence of sentences formed by one side of a parallel corpus. The Suffix Array is then the sorted list of all the suffixes of all the sentences in the sequence. Suffixes may be represented as a pair of integer (*index of the sentence*, *position in the sentence*) or again as a pointer (an example using integer pairs is shown on Figure 1).

We can implement the Suffix Array so that, from a suffix, we can determine the index of the sentence to which it belongs (the computational cost of this is marginal in practical cases and will be neglected). We can now compute pf and pl for a substring S such as previously, and retrieve the sentence indexes corresponding to every suffix between positions pf and pl in the Suffix Array, This allow us to create an "*occurrence vector*": a mapping between sentence indexes and the number of occurrences of S in those sentences. This operation takes $O(pl-pf)$, that is $O(N(S))$. (Figure 1. shows an occurrence vector for the substring "red car")

We can now efficiently compute the co-occurrence counts of two substrings S_1 and S_2 in a parallel corpus.

We compute beforehand the two Suffix Arrays for the 2 sides of the parallel corpus. We can then compute two occurrence vectors V_1 and V_2 for S_1 and S_2 in $O(N(S_1)+|S_1|*log(T_1))$ and $O(N(S_2)+|S_2|*log(T_2))$ respectively.

A small monolingual corpus	
index	sentence
1	The red car is here
2	I saw a blue car
3	I saw a red car

Occurrence Vector of "**red car**"	
index	nbOcc
1	1
2	0
3	1

Suffix Array		
Array index	Suffix Position	Suffix
0	2,3	a blue car
1	3,4	a red car
2	2,4	blue car
3	2,6	car
4	3,5	car
5	1,3	car is here
6	1,5	here
7	2,1	I saw a blue car
8	1,1	I saw a red car
9	1,4	is here
10	**1,2**	**red car is here**
11	**3,5**	**red car**
12	2,2	saw a blue car
13	3,3	saw a red car
14	1,1	The red car is here

Figure 1. A small corpus, the corresponding suffix array, and an occurrence vector

With a good implementation, we can use these two vectors to obtain $N(S_1,S_2)$ in $O(min(size(V_1),size(V_2)))$, that is $O(min(N(S_1),N(S_2)))$.

Hence we can compute $NbCoOcc(S_1,S_2)$ for any substring pair (S_1,S_2) in $O(N(S_2)+|S_2|*log(T_2)+N(S_1)+|S_1|*log(N_1)))$. This is much better than a naive approach that takes $O(T_1*T_2)$ by going through the whole corpus. Besides, some simple optimizations will substantially improve the average performances.

2.5 Some Important Optimizations

There are two ways to improve performances when using the previous method for co-occurrences computing.

Firstly, we won't compute co-occurrences for any substrings at random. Typically, in the algorithm described in the following part, we compute $N(S_1,S_2)$ for every substring pairs in a given bi-sentence. So we will compute the occurrence vector of a substring only once per sentence.

Secondly, the time taken to retrieve the co-occurrence count of two substrings S_1 and S_2 is more or less proportional to their frequency in the corpus. This is a problem for the average performance: the most frequent substrings will be the one that take longer to compute. This suggests that by caching the occurrence vectors of the most frequent substrings (as well as their co-occurrence counts), we might expect a good improvement in performance. (We will see in the next sub-section that caching the 200 most frequent substrings is sufficient to multiply the average speed by a factor of 50)

2.6 Practical Evaluation of the Performances

We will now test the computational practicality of our method. For this evaluation, we will consider the English-Japanese BTEC corpus (170,000 bi-sentences, 12MB), and the English-French Europarl corpus (688,000 bi-sentences, 180 MB). We also want to apply our algorithm to western languages at the character level. However, working at a character level multiply the size of the suffix array by about 5, and increase the size of the cached vectors as well. So, because of memory limitations, we extracted a smaller corpus from the Europarl one (100,000 bi-sentences, 20MB) for experimenting on characters substrings.

The base elements we will choose for our substrings will be: word/characters for the BTEC, word/word for the bigger EuroParl, and word/characters for the smaller EuroParl. We computed the co-occurrence counts of every substrings pair in a bi-sentence for the 100 first bi-sentences of every corpus, on a 2.5GHz x86 computer. We give the average figures for different corpora and caching strategies.

Corpus	Cache (cached substrg)	Allocated Memory (MB)	CoOcc computed (per sec.)	bisentences processed (per sec.)
BTEC	0	22	7k	1.2
BTEC	200	120	490k	85
EuroParl	0	270	3k	0.4
EuroParl	400	700	18k	1.2
Small EuroParl	0	100	4k	0.04
Small EuroParl	400	300	30k	0.3

These results are good enough and show that the algorithm we are going to introduce is not computationally impracticable. The cache allows an interesting trade-off between the performances and the used memory. We note that the proportional speedup depends on the corpus used. We did not investigate this point, but the different sizes of corpora (inducing different average *occurrence vectors* sizes), and the differences in the frequency distribution of words and characters are probably the main factors.

3 Sub-sentential alignment

3.1 The General Principle

Given two substrings S_1 and S_2, we can use their occurrence and co-occurrence counts to compute a correlation coefficient (such as the chi-square statistic, the point-wise mutual information or the Dice coefficient).

The basic principle of our sub-sentential alignment algorithm will simply be to compute a correlation coefficient between every substring in a bi-sentence, and align the substrings with the highest correlation. This idea needs, however, to be refined.

First, we have to take care of the indirect association problem. The problem, which was described in (Melamed, 1997) in a word-to-word alignment context, is as follows: if e_1 is the translation of f_1 and f_2 has a strong monolingual association with f_1, e_1 and f_2 will also have a strong correlation. Melamed assumed that indirect associations are weaker than direct ones, and provided a Competitive Linking Algorithm that does not allow for a word already aligned to be linked to another one. We will make the same assumption and apply the same solution. So our algorithm will align the substring pairs with the highest correlation first, and will forbid the subsequent alignment of substrings having a part in common with an already aligned substring. A side-effect of this procedure is that we will be constrained to produce a single segmentation on both sentences and a single alignment between the components of this segmentation. According to the application, this might be what we are looking for or not. But it must be noted that, most of the time, alignments with various granularities are possible, and we will only be able to find one of them. We will discuss the issue of the granularity of the alignment in part **3.3**.

Besides, our approach implicitly considers that the translation of a substring is a substring (there are no discontinuities). This is of course not the case in general (for example, the English word "*not*" is usually translated in French by "*ne...pas*"). However, there is most of the time a granularity of alignment at which there is no discontinuity in the alignment components.

Also, it is frequent that a word or a sequence of words in a sentence has no equivalent in the opposite sentence. That is why it will not be mandatory for our algorithm to align every element of the sentences at all cost. If, at any point, the substrings that are yet to be linked have correlation coefficients below a certain threshold, the algorithm will not go further.

So, the algorithm can be described as follow:

1- Compute a correlation coefficient for all the substrings pairs in e and f and mark all the elements in e and f as free.

2- Among the substrings which contain only free element, find the pair with the highest correlation. If this correlation is not above a certain threshold, end the algorithm. Else, output a link between the substrings of the pair.

3- Mark all the elements belonging to the linked pair as non-free.

4- Go back to 2

It should be noted that correlation coefficients are only meaningful data is sufficiently available; but many substrings will appear only a couple of times in the corpus. That is why, in our experiments we have set to zero the correlation coefficient of substring pairs that co-occur less than 5 times (this might be a bit conservative, but the BTEC corpus we used being very redundant, it was not too much of a restriction).

3.2 Giving a preference to bigger alignments.

A problem that arose in applying the previous algorithm is a tendency to link incomplete substrings. Typically, this happen when a substring S_1 can be translated by two substrings S_2 and S_2', S_2 and S_2' having themselves a common substring. S_1 will then be linked to the common part of S_2 and S_2'. For example, the English word "museum" has two Japanese equivalents: 博物館 and 美術館. In the BTEC corpus, the common part (館) will have a stronger association with "museum", and so will be linked instead of the correct substring (博物館 or 美術館).

To prevent this problem, we have tried to modify the correlation coefficients so that they slightly penalize shorter alignment. Precisely, for a substring pair (S_1,S_2), we define its area as *"length of* S_1*"*"length of* S_2*"*. We then multiply the Dice coefficient by $area(S_1,S_2)$ and the chi-square coefficient by $log(area(S_1,S_2)+1)$. These formulas are very empiric, but they created a considerable improvement in our experimental results.

Linking the bigger parts of the sentences first has another interesting effect: bigger substrings present less ambiguity, and so linking them first may prevent further ambiguities to arise. For example, with the bi-sentence *"the cat on the wall"/"le chat sur le mur"*. Each *"the"* in the English sentence will have the same correlation with each *"le"* in the French sentence, and so the algorithm cannot determine which *"the"* correspond to which *"le"*. But if, for example *"the cat"* has been previously linked to *"le chat*", there is no more ambiguity.

We mentioned previously the issue of the granularity of alignments. These "alignment size penalties" could also be used to tune the granularity of the alignment produced.

3.3 Experiments and Evaluations

Although we made some tests to confirm that computation time did not prevent our algorithm to work with bigger corpus such as the EuroParl corpus, we have until now limited deeper experiments to the Japanese-English BTEC Corpus.

That is why we will only present results for this corpus. For comparison, we re-implemented the ISA (Integrated Segmentation Alignment) algorithm described in (Zhang, Vogel and Waibel, 2003). This algorithm is interesting because it is somehow similar to our own approach, in that it can be seen as a generalization of Melamed's Competitive Linking Algorithm. It is also fairly easy to implement. A comparison with the joint probability model of Marcu and Wong (which can also work at the phrase/substring level) would have also been very interesting, but the difficulty of implementing and adapting the algorithm made us delay the experiment.

After trying different settings, we chose to use chi-square statistic as the correlation coefficient for the ISA algorithm, and the dice coefficient for our own algorithm. ISA settings as well as the "alignment size penalties" of our algorithm were also tuned to give the best results possible with our test set. For our algorithm, we considered word-substrings for English and characters substrings for Japanese. For the ISA algorithm, we pre-segmented the Japanese corpus, but also tried to apply it directly to Japanese by considering characters as words.

Estimating the quality of an alignment is not an easy thing. We tried to compute a precision and a recall score in the following manner. Precision was such that:

$$Precision = \frac{Nb\ of\ correct\ links}{Nb\ of\ outputted\ links}$$

Correct link are counted by manual inspection of the results. Appreciating what is a correct link is subjective; especially here, where we consider many-words-to-many-characters links. Overall, the evaluation was pretty indulgent, but tried to be consistent, so that the comparison would not be biased.

Computing recall is more difficult: for a given bi-sentence, multiple alignments with different granularities are possible. As we are only trying to output one of these alignments, we cannot define easily a "gold standard". What we did was to count a missed link for every element that was not linked correctly and could have been. We then compute a recall measure such that:

$$Recall=\frac{Nb\ of\ correct\ links}{Nb\ of\ correct\ links + Nb\ of\ missed\ links}.$$

These measures are not perfect and induce some biases in the evaluation (they tend to favor algorithms aligning bigger part of the sentence, for example), but we think they still give a good summary of the results we have obtained so far.

As can be seen in the following table, our algorithm performed quite well. We are far from the results obtained with a pre-segmentation, but considering the simplicity of this algorithm, we think these results are encouraging and justify our initial ideas. There is still a lot of room for improvement: introducing a n-gram language model, using multiple iterations to re-estimate the correlation of the substrings...

That is why we are pretty confident that we can hope to compete in the end with algorithms using pre-segmentation.

	Precision	Recall
Our algorithm (w/o segmentation)	78%	70%
ISA (w/o segmentation)	55%	55%
ISA + segmentation	98%	95%

Also, although we did not make any thorough evaluation, we also applied the algorithm to a subset of the Europarl corpus (cf. **2.6**), where characters where considered the base unit for French. The alignments were mostly satisfying (seemingly better than with the BTEC). But hardly any sub-word alignments were produced. Some variations on the ideas of the algorithm, however, allowed us to get interesting (if infrequent) results. For example, in the pair ('I would like'/ 'Je voudrais'), 'would' was aligned with 'rais' and 'voud' with 'like'.

4 Conclusion and future work

In this paper we presented both a method for accessing the co-occurrences count for any substring pair in a parallel corpus and an algorithm taking advantage of this method to create subsentential alignments in such a corpus.

We showed our co-occurrence counting method performs well with corpus commonly used in Statistical Machine Translation research, and so we think it can be a useful tool for the statistical processing of parallel corpora.

Our phrase level alignment algorithm gave encouraging results, especially considering there are many possibilities for further improvement.

In the future, we will try to improve the algorithm as well as perform more extensive evaluations on different language pairs.

References

Ralph Brown. 1997. *Automated Dictionary Extraction for Knowledge-Free Example Based Translation*, Proceedings of the 7th International Conference on Theoretical and Methodological Issues in Machine Translation, pp. 111-118, Santa-Fe, July 1997.

Chris Callison-Burch, Colin Bannard and Josh Scroeder. 2005. *Scaling Phrase-Based Statistical Machine Translation to Larger Corpora and Longer Phrases*, Proceedings of 43rd Conference of the Association for Computational Linguistics (ACL 05), Ann Arbor, USA, 2005.

Philipp Koehn. 2003. *Europarl: A Multilingual Corpus for Evaluation of Machine Translation*, Draft,Unpublished.

Manber and Myers. 1993. *Suffix Array: A New Method for On-Line String Searches*, SIAM Journal on Computing, 22(5):935-948.

Daniel Marcu, William Wong. 2002. *A Phrase-Based, Joint Probability Model for Statistical Machine Translation*, Proceedings of the Conference on Empirical Methods in Natural Language Processing , Philadelphia, USA, July 2002.

Dan Melamed. 1997. *A Word-to-Word Model of Translational Equivalence*, Proceedings of 35th Conference of the Association for Computational Linguistics (ACL 97), Madrid, Spain, 1997.

Franz Joseph Och, Christophe Tillmann, Hermann Ney. 1999. *Improved Alignment Models for Statistical Machine Translation*. Proceedings of the joint conference of Empirical Methods in Natural Language Processing and Very Large Corpora, pp 20-28, University Of Maryland,

Jia Xu, Richard Zens., Hermann Ney. 2004. *Do We Need Chinese Word Segmentation for Statistical Machine Translation?*, Proceedings of the 3rd SIGHAN Workshop on Chinese Language Learning, Barcelona, Spain, pp. 122-128 , July 2004

Ying Zhang, Stephan Vogel and Alex Waibel. 2003. *Integrated Phrase Segmentation and Alignment algorithm for Statistical Machine Translation*, Proceedings of International Conference on Natural Language Processing and Knowledge Engineering, Beijing,China., October 2003

Annotation Schemes and their Influence on Parsing Results

Wolfgang Maier
Seminar für Sprachwissenschaft, Universität Tübingen
Wilhelmstr. 19, 72074 Tübingen, Germany
wmaier@sfs.uni-tuebingen.de

Abstract

Most of the work on treebank-based statistical parsing exclusively uses the Wall-Street-Journal part of the Penn treebank for evaluation purposes. Due to the presence of this quasi-standard, the question of to which degree parsing results depend on the properties of treebanks was often ignored. In this paper, we use two similar German treebanks, TüBa-D/Z and NeGra, and investigate the role that different annotation decisions play for parsing. For these purposes, we approximate the two treebanks by gradually taking out or inserting the corresponding annotation components and test the performance of a standard PCFG parser on all treebank versions. Our results give an indication of which structures are favorable for parsing and which ones are not.

1 Introduction

The Wall-Street-Journal part (WSJ) of the Penn Treebank (Marcus et al., 1994) plays a central role in research on statistical treebank-based parsing. It has not only become a standard for parser evaluation, but also the foundation for the development of new parsing models. For the English WSJ, high accuracy parsing models have been created, some of them using extensions to classical PCFG parsing such as lexicalization and markovization (Collins, 1999; Charniak, 2000; Klein and Manning, 2003). However, since most research has been limited to a single language (English) and to a single treebank (WSJ), the question of how portable the parsers and their extensions are across languages and across treebanks often remained open.

Only recently, there have been attempts to evaluate parsing results with respect to the properties and the language of the treebank that is used. Gildea (2001) investigates the effects that certain treebank characteristics have on parsing results, such as the distribution of verb subcategorization frames. He conducts experiments on the WSJ and the Brown Corpus, parsing one of the treebanks while having trained on the other one. He draws the conclusion that a small amount of matched training data is better than a large amount of unmatched training data. Dubey and Keller (2003) analyze the difficulties that German imposes on parsing. They use the NeGra treebank for their experiments and show that lexicalization, while highly effective for English, has no benefit for German. This result motivates them to create a parsing model for German based on sister-head-dependencies. Corazza et al. (2004) conduct experiments with model 2 of Collins' parser (Collins, 1999) and the Stanford parser (Klein and Manning, 2003) on two Italian treebanks. They report disappointing results which they trace back to the different difficulties of different parsing tasks in Italian and English and to differences in annotation styles across treebanks.

In the present paper, our goal is to determine the effects of different annotation decisions on the results of plain PCFG parsing without extensions. Our motivation is two-fold: first, we want to present research on a language different from English, second, we want to investigate the influences of annotation schemes via a realistic comparison, i.e. use two different annotation schemes. Therefore, we take advantage of the availability of two similar treebanks of German, TüBa-D/Z (Telljohann et al., 2003) and NeGra (Skut et al., 1997). The strategy we adopt extends Kübler

(2005). Treebanks and their annotation schemes respectively are compared using a stepwise approximation. Annotation components corresponding to certain annotation decisions are taken out or inserted, submitting each time the resulting modified treebank to the parser. This method allows us to investigate the role of single annotation decisions in two different environments.

In section 2, we describe the annotation of both treebanks in detail. Section 3 introduces the methodology used. In section 4, we describe our experimental setup and discuss the results. Section 5 presents a conclusion and plans for future work.

2 The Treebanks: TüBa-D/Z and NeGra

With respect to treebanks, German is in a privileged position. Various treebanks are available, among them are two similar ones: NeGra (Skut et al., 1997), from Saarland University at Saarbrücken and TüBa-D/Z (Telljohann et al., 2003), from the University of Tübingen. NeGra contains about 20,000 sentences, TüBa-D/Z about 15,000, both consist of newspaper text. In both treebanks, predicate argument structure is annotated, the core principle of the annotation being its theory independence. Terminal nodes are labeled with part-of-speech tags and morphological labels, non-terminal nodes with phrase labels. All edges are labeled with grammatical functions. Annotation was accomplished semi-automatically with the same software tools.

The main difference between the treebanks is rooted in the partial free word order of German sentences: the positions of complements and adjuncts are of great variability. This leads to a high number of discontinuous constituents, even in short sentences. An annotation scheme for German must account for that. NeGra allows for crossing branches, thereby giving up the context-free backbone of the annotation. With crossing branches, discontinuous constituents are not a problem anymore: all children of every constituent, discontinuous or not, can always be grouped under the same node. The inconvenience of this method is that the crossing branches must be resolved before the treebank can be used with a (PCFG) parser. However, this can be accomplished easily by reattaching children of discontinuous constituents to higher nodes.

TüBa-D/Z uses another mechanism to account for the free word order. Above the phrase level, an additional layer of annotation is introduced. It consists of topological fields (Drach, 1937; Höhle, 1986). The concept of topological fields is widely accepted among German grammarians. It reflects the empirical observation that German has three possible sentence configurations with respect to the position of the finite verb. In its five fields (initial field, left sentence bracket, middle field, right sentence bracket, final field), verbal material generally resides in the two sentence brackets, while the initial field and the middle field contain all other elements. The final field contains mostly extraposed material. Since word order variations generally do not cross field boundaries, with the model of topological fields, the free word order of German can be accounted for in a natural way.

On the phrase level, the treebanks show great differences, too. NeGra does not allow for any intermediate ("bar") phrasal projections. Additionally, no unary productions are allowed. This results in very flat phrases: pre- and postmodifiers are attached directly to the phrase, nominal subjects are attached directly to the sentence, nominal material within PPs doesn't project to NPs, complex (non-coordinated) NPs remain flat. TüBa-D/Z, on the contrary, allows for "deep" annotation. Intermediate productions and unary productions are allowed and extensively used.

To illustrate the annotation principles, the figures 1 and 2 show the annotation of the sentences (1) and (2) respectively.

(1) Darüber muß nachgedacht werden.
About-that must tought be
'This must be tought about.'

(2) Schillen wies dies gestern zurück:
Schillen rejected that yesterday VPART
'Schillen rejected that yesterday.'

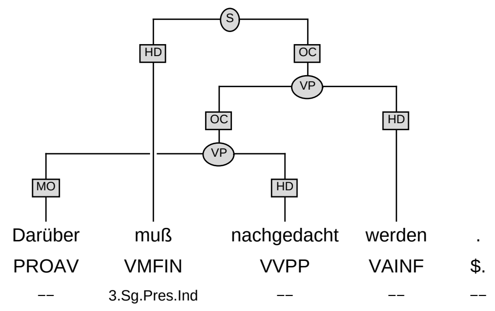


Figure 1: A NeGra tree

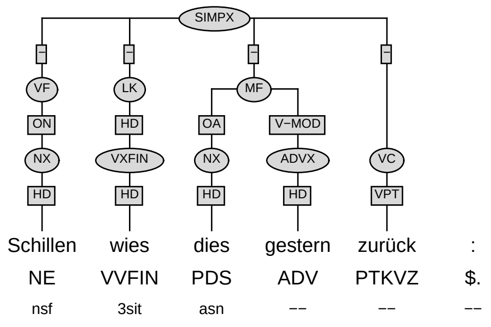


Figure 2: A TüBa-D/Z tree

3 Treebanks, Parsing, and Comparisons

Our goal is to determine which components of the annotation schemes of TüBa-D/Z and NeGra have which influence on parsing results. A direct comparison of the parsing results shows that the TüBa-D/Z annotation scheme is more appropriate for PCFG parsing than NeGra's (see tables 2 and 3). However, this doesn't tell us anything about the role of the subparts of the annotation schemes.

A first idea for a more detailed comparison could be to compare the results for different phrase types. The problem is that this would not give meaningful results. NeGra noun phrases, e.g., cover a different set of constituents than TüBa-D/Z noun phrases, due to NeGra's flat annotation and avoidance of annotation of unary NPs. Furthermore, both annotation schemes contain categories not contained in the other one. There are, e.g., no categories in NeGra that correspond to TüBa-D/Z's field categories, while in TüBa-D/Z, there are no categories equivalent to NeGra's categories for coordinated phrases or verb phrases.

We therefore pursue another approach. We use a method introduced by Kübler (2005) to investigate the usefulness of different annotation components for parsing. We gradually modify the treebank annotations in order to approximate the annotation style of the treebanks to one another. This is accomplished by taking out or inserting certain components of the annotation. For our treebanks, this generally results in reduced structures for TüBa-D/Z and augmented structures for NeGra. Table 1 presents three measures that capture the changes between each of the modifications. The average number of child nodes of nonterminal nodes shows the degree of flatness of the annotation on phrase level. Here, the unmodified NeGra consequently shows the highest values. The average tree height relates directly to the number of annotation hierarchies in the tree. Here, the unmodified TüBa-D/Z has the highest values.

4 Experimental Setup

For our experiments, we use `lopar` (Schmid, 2000), a standard PCFG parser. We read the grammar and the lexicon directly off the trees together with their frequencies. The parser is given the gold POS tagging to avoid parsing errors that are caused by wrong POS tags. Only sentences up to a length of 40 words are considered due to memory limitations.

Traditionally, most of the work on WSJ uses the same section of the treebank for testing. However, for our aims, this method has a shortcoming: since both treebanks consist of text created by different authors, linguistic phenomena are not evenly distributed over the treebank. When using a whole section as test set, some phenomena may only occur there and thus not occur in the grammar. To reduce data sparseness, we use another test/training-set split for the treebanks and their variations. Each 10th sentence is put into the test set, all other sentences go into the training set.

4.1 Preprocessing the Treebanks

Since we want to read the grammars for our parser directly off the treebanks, preprocessing of the treebanks is necessary due to the non-context-free nature of the original annotation. In both treebanks, punctuation is not included in the trees, furthermore, sentence splitting in both treebanks does not always coincide with the linguistic notion of a sentence. This leads to sentences consisting of several unconnected trees. All nodes in a sentence, i.e. the roots and the punctation, are grouped by a virtual root node, which may cause crossing branches. Furthermore, the NeGra annotation scheme allows for crossing branches for linguistic reasons, as described in section 2. All of the crossing branches have to be removed before parsing.

The crossing branches caused by the NeGra annotation scheme are removed with a small program by Thorsten Brants. It attaches some of the children of discontinuous constituents to higher nodes. The virtual root node is made continuous by attaching all punctuation to the highest possible location in the tree. Pairs of parenthesis and quotation marks are preferably attached to

	NeGra	*NE_fi.*	*NE_NP*	*NE_tr.*	TüBa	Tü_NF	Tü_NU	Tü_f	Tü_f_NU	Tü_f_NU_NF
N/T	0.41	0.70	0.50	0.41	1.21	0.89	0.54	1.00	0.42	0.35
μ D/N	2.92	2.22	2.59	2.92	1.61	1.89	2.53	1.83	2.93	3.35
μ H(T)	4.86	5.81	5.16	4.68	6.88	5.68	5.45	5.94	4.72	4.15

Table 1: Properties of the treebank modifications[1]

the same node, to avoid low-frequent productions in the grammar that only differ by the position of parenthesis marks on their right hand side.

4.2 Results of the Comparison

We use the standard parseval measures for the evaluation of parser output. They measure the percentage of correctly parsed constituents, in terms of precision, recall, and F-Measure. The parser output of each modified treebank version is evaluated against the correspondingly modified test set. Unparsed sentences are fully included in the evaluation.

NeGra. Along with the unmodified treebank, two modifications of NeGra are tested. Both of them introduce annotation components present in TüBa-D/Z but not in NeGra. In the first one, *NE_fi*, we add an annotation layer of **topological fields**[2], as existing in TüBa-D/Z. The precision value benefits the most from this modification. When parsing without grammatical functions, it increases about 6,5%. When parsing with grammatical functions, it increases about 14%. Thus, the additional rules provided by a topological field level that groups phrases below the clausal level are favorable for parsing. The average number of crossing brackets per sentence increases, which is due to the fact that there are simply more brackets to create.

A detailed evaluation of the results for node categories shows that the new field categories are easy to recognize (e.g. LF gets 97.79 F-Measure). Nearly all categories have a better precision value. However, the F-Measure for VPs is low (only 26.70 while 59.41 in the unmodified treebank), while verb phrases in the unmodified TüBa-D/Z (see below) are recognized with nearly 100 points F-Measure. The problem here is the following. In the original NeGra annotation, a verb and its complements are grouped under the same VP. To preserve as much of the annotation as possible, the topological fields are inserted *below* the VP (complements are grouped by a middle field node, the verb complex by the right sentence bracket). Since this way, the phrase node VP resides above the field level, it becomes difficult to recognize.

In the second modification, *NE_NP*, we approximate NeGra's PPs to TüBa-D/Z's by grouping all **nominal material below the PPs to separate NPs**. This modification gives us a small benefit in terms of precision and recall (about 2-3%). Although there are more brackets to place, the number of crossing parents increases only slightly, which can be attributed to the fact that below PPs, there is no room to get brackets wrong.

We finally parse a version of NeGra where for each node movement during the resolution of crossing edges, a **trace label** was created in the corresponding edge (*NE_tr*). Although this brings the treebank closer to the format of TüBa-D/Z, the results get even worse than in the version without traces. However, the high number of unparsed sentences indicates that the result is not reliable due to data sparseness.

	NeGra	*NE_fi.*	*NE_NP*	*NE_tr.*
	without grammatical functions			
cross. br.	1.10	1.67	1.14	—
lab. prec.	68.14%	74.96%	70.43%	—
lab. rec.	69.98%	70.37%	72.81%	—
lab. F_1	69.05	72.59	71.60	—
not parsed	1.00%	0.10%	0.15%	—
	with grammatical functions			
cross. br.	1.10	1.21	1.27	1.05
lab. prec.	52.67%	67.90%	59.77%	51.81%
lab. rec.	52.17%	65.18%	60.36%	49.19%
lab. F_1	52.42	66.51	60.06	50.47
not parsed	12.90%	1.66%	9.88%	16.01%

Table 2: Parsing NeGra: Results

TüBa-D/Z. Apart from the original treebank, we test six modifications of TüBa-D/Z. In each of the modifications, annotation material is removed in order to obtain NeGra-like structures. Since they are equally absent in NeGra, we delete the annotation of **topological fields** in the first modification, *Tü_NF*. This results in small losses.

[1]explanation: N/T = node/token ratio, μ D/N = average number of daughters of non-terminal nodes, μ H(T) = average tree height

[2]We are grateful to the DFKI Saarbrücken for providing us with the topological field annotation.

	TüBa	Tü_NF	Tü_NU	Tü_flat	Tü_f_NU	Tü_f_NU_NF
	without grammatical functions					
crossing brackets	2.21	1.82	1.67	1.04	0.80	1.03
labeled precision	87.39%	86.31%	79.97%	86.22%	75.18%	63.05%
labeled recall	83.57%	83.43%	78.52%	85.41%	76.11%	66.86%
labeled F-Measure	85.44	84.85	79.24	85.81	75.64	64.90
not parsed	0.07%	0.07%	2.45%	0.07%	2.99%	6.87%
	with grammatical functions					
crossing brackets	1.84	1.82	1.79	0.98	1.01	1.12
labeled precision	76.99%	68.55%	63.71%	76.93%	58.91%	45.15%
labeled recall	75.30%	68.40%	62.79%	77.21%	58.92%	44.76%
labeled F-Measure	76.14	68.47	63.25	77.07	58.92	44.96
not parsed	0.07%	0.27%	4.49%	0.07%	7.21%	17.76%

Table 3: Parsing TüBa-D/Z: Results

A closer look at category results shows that losses are mainly due to categories on the clausal level; structures within fields do not deteriorate. Field categories are thus especially helpful for the clausal level.

In the second modification of TüBa-D/Z, *Tü_NU*, **unary nodes** are collapsed with the goal to get structures comparable to NeGra's. As the figures show, the unary nodes are very helpful, the F-Measure drops about 6 points without them. The number of crossing brackets also drops, along with the total number of nodes. When parsing with grammatical functions, taking out unary productions has a detrimental effect, F-Measure drops about 13 points. A plausible explanation could be data sparseness. 32.78% of the rules that the parser needs to produce a correct parse don't occur in the training set.

An evaluation of the results for the different categories shows that all major phrase categories loose both in precision and recall. Since field nodes are mostly unary, many of them disappear, but most of the middle field nodes stay because they generally contain more than one element. However, their recall drops about 10%. Supposedly it is more difficult for the parser to annotate the middle field "alone" without the other field categories.

We also test a version of TüBa-D/Z with **flattened phrases** that mimic NeGra's flat phrases, *Tü_flat*. With this treebank version, we get results very similar to those of the unmodified treebank. The F-Measure values are slightly higher and the parser produces less crossing brackets. A single category benefits the most from this treebank modification: EN-ADD, its F-Measure rising about 45 points. It was originally introduced as a marker for named entities, which means that it has no specific syntactic function. In the TüBa-D/Z version with flattened phrases, many of the nominal nodes below EN-ADD are taken out, bringing EN-ADD closer to the lexical level. This way, the category has more meaningful context and therefore produces better results.

Furthermore, we test combinations of the modifications. Apart from the average tree height, the dimensions of TüBa-D/Z with **flattened phrases and without unary productions** (*Tü_f_NU*) resemble those of the unmodified NeGra treebank, which indicates their similarity. Nevertheless, parser results are worse on NeGra. This indicates that TüBa-D/Z still benefits from the remaining field nodes. The number of crossing branches is the lowest in this treebank version.

In the last modification that **combines all modifications** made before (*TÜ_f_NU_NF*), as expected, all values drop dramatically. F-Measure is about 5 points worse than with the unmodified NeGra treebank.

POS tagging. In a second round, we investigate the benefits that gold POS tags have when making them available in the parser input. We repeat all experiments without giving the parser the perfect tagging.

This leads to higher time and space requirements during parsing, caused by the additional tagging step. With TüBa-D/Z, NeGra, and all their modifications, the F-Measure results are about 3-5 points worse when parsing with grammatical functions. When parsing without them, they drop 3-6 points. We can determine two exceptions: TüBa-D/Z with flattened phrases, where the F-Score drops more than 9 points when parsing with grammatical functions, and the TüBa-D/Z version with all modifications combined, where F-Score drops only a little less than 2 points. The behavior of the flattened TüBa-D/Z relates directly to the fact that the categories that loose the most without gold POS tags are phrase categories (particularly infinite VPs and APs). They are directly conditioned on the POS tagging and thus behave accordingly to its quality. For the TüBa-D/Z version with all modifications combined, one could argue that the results are not reliable because of data sparseness, which is confirmed by the high number of unparsed sentences in this treebank version. However, in all cases, less crossing brackets are produced.

To sum up, obviously, it is more difficult for the parser to build a parse tree onto an already existing layer of POS-tagging. This explains the bigger number of unparsed sentences. Nevertheless, in terms of F-Score, the parsing results profit visibly from the gold POS tagging.

5 Conclusions and Outlook

We presented an analysis of the influences of the particularities of annotation schemes on parsing results via a comparison of two German treebanks, NeGra and TüBa-D/Z, based on a stepwise approximation of both treebanks. The experiments show that as treebanks are approximated, the parsing results also get closer. When annotation structure is deleted in TüBa-D/Z, the number of crossing brackets drops, but F-Measure drops, too. When annotation structure is added in NeGra, the contrary happens. We can conclude that, being interested in good F-Measure results, the deep TüBa-D/Z structures are more appropriate for parsing than NeGra's flat structures. Moreover, we have observed that it is beneficial to provide the parser with the gold POS tags at parsing time. However, we see that especially when parsing with grammatical functions, data sparseness becomes a serious problem, making the results less reliable.

Seen in the context of a parse tree, the expansion probability of a PCFG rule just covers a subtree of height 1. This is a clear deficiency of PCFGs since this way, e.g., the expansion probability of a VP is independent of the choice of the verb. Our future work will start at this point. We will conduct further experiments with the Stanford Parser (Klein and Manning, 2003) which considers broader contexts in its probability. It uses markovization to reduce horizontal context (right hand sides of rules are broken up) and add vertical context (rule probabilities are conditioned on (grand-)parent-node information). This way, we expect further insights in NeGra's an TüBa-D/Z's annotation schemes.

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*. Ph.D. thesis, University of Pennsylvania.

Anna Corazza, Alberto Lavelli, Giorgio Satta, and Roberto Zanoli. 2004. Analyzing an Italian treebank with state-of-the-art statistical parsers. In *Proceedings of the 3^{rd} Workshop on Treebanks and Linguistic Theories (TLT 2004)*.

Erich Drach. 1937. *Grundgedanken der deutschen Satzlehre*. Diesterweg, Frankfurt/Main.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using sisterhead dependencies. In *Proceedings of ACL 2003*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of EMNLP 2001*.

Tilman Höhle. 1986. Der Begriff "Mittelfeld", Anmerkungen ber die Theorie der topologischen Felder. In *Akten des Siebten Internationalen Germanistenkongresses 1985*, Göttingen, Germany.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL 2003*.

Sandra Kübler. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005*.

Mitchell P. Marcus, Grace Kim, Marry Ann Marcinkiewicz, Robert MacIntyre, Ann Biew, Mark Freguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the 1994 Human Language Technology Workshop, HLT 94, Plainsboro, NJ*.

Helmut Schmid. 2000. LoPar: Design and implementation. Technical report, Universität Stuttgart, Germany.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP 1997*.

Heike Telljohann, Erhard W. Hinrichs, and Sandra Kübler, 2003. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Modeling Human Sentence Processing Data with a Statistical Parts-of-Speech Tagger

Jihyun Park
Department of Linguisitcs
The Ohio State University
Columbus, OH, USA
park@ling.ohio-state.edu

Abstract

It has previously been assumed in the psycholinguistic literature that finite-state models of language are crucially limited in their explanatory power by the locality of the probability distribution and the narrow scope of information used by the model. We show that a simple computational model (a bigram part-of-speech tagger based on the design used by Corley and Crocker (2000)) makes correct predictions on processing difficulty observed in a wide range of empirical sentence processing data. We use two modes of evaluation: one that relies on comparison with a control sentence, paralleling practice in human studies; another that measures probability drop in the disambiguating region of the sentence. Both are surprisingly good indicators of the processing difficulty of garden-path sentences. The sentences tested are drawn from published sources and systematically explore five different types of ambiguity: previous studies have been narrower in scope and smaller in scale. We do not deny the limitations of finite-state models, but argue that our results show that their usefulness has been underestimated.

1 Introduction

The main purpose of the current study is to investigate the extent to which a probabilistic part-of-speech (POS) tagger can correctly model human sentence processing data. Syntactically ambiguous sentences have been studied in great depth in psycholinguistics because the pattern of ambiguity resolution provides a window onto the human sentence processing mechanism (HSPM). *Prima facie* it seems unlikely that such a tagger will be adequate, because almost all previous researchers have assumed, following standard linguistic theory, that a formally adequate account of recursive syntactic structure is an essential component of any model of the behaviour. In this study, we tested a bigram POS tagger on different types of structural ambiguities and (as a sanity check) to the well-known asymmetry of subject and object relative clause processing.

Theoretically, the garden-path effect is defined as processing difficulty caused by reanalysis. Empirically, it is attested as comparatively slower reading time or longer eye fixation at a disambiguating region in an ambiguous sentence compared to its control sentences (Frazier and Rayner, 1982; Trueswell, 1996). That is, the garden-path effect detected in many human studies, in fact, is measured through a "comparative" method.

This characteristic of the sentence processing research design is reconstructed in the current study using a probabilistic POS tagging system. Under the assumption that larger probability decrease indicates slower reading time, the test results suggest that the probabilistic POS tagging system can predict reading time penalties at the disambiguating region of garden-path sentences compared to that of non-garden-path sentences (i.e. control sentences).

2 Experiments

A Hidden Markov Model POS tagger based on bigrams was used. We made our own implementation to be sure of getting as close as possible to the design of Corley and Crocker (2000). Given a word string, $w_0, w_1, \cdots, w_n$, the tagger calculates the probability of every possible tag path,

$t_0, \cdots, t_n$. Under the Markov assumption, the joint probability of the given word sequence and each possible POS sequence can be approximated as a product of conditional probability and transition probability as shown in (1).

(1) $P(w_0, w_1, \cdots, w_n, t_0, t_1, \cdots, t_n)$

$\approx \Pi_{i=1}^{n} P(w_i|t_i) \cdot P(t_i|t_{i-1})$, where $n \geq 1$.

Using the Viterbi algorithm (Viterbi, 1967), the tagger finds the most likely POS sequence for a given word string as shown in (2).

(2) $\arg\max P(t_0, t_1, \cdots, t_n|w_0, w_1, \cdots, w_n, \mu)$.

This is known technology, see Manning and Schütze (1999), but the particular use we make of it is unusual. The tagger takes a word string as an input, outputs the most likely POS sequence and the final probability. Additionally, it presents accumulated probability at each word break and probability re-ranking, if any. Probability re-ranking occurs when a previously less preferred POS sequence is more favored later. Note that the running probability at the beginning of a sentence will be 1, and will keep decreasing at each word break since it is a product of conditional probabilities.

We tested the predictability of the model on empirical reading data with the probability decrease and the presence or absence of probability re-ranking. Probability re-ranking occurs when a less preferred POS sequence is selected later over a temporarily favored sequence. Adopting the standard experimental design used in human sentence processing studies, where word-by-word reading time or eye-fixation time is compared between an experimental sentence and its control sentence, this study compares probability at each word break between a pair of sentences. Comparatively faster drop of probability is expected to be a good indicator of comparative processing difficulty. Probability re-ranking, which is a simplified model of the reanalysis process assumed in many human studies, is also tested as another indicator of garden-path effect. Probability re-ranking will occur when an initially dispreferred POS subsequence becomes the preferred candidate later in the parse, because it fits in better with later words.

The model parameters, $P(w_i|t_i)$ and $P(t_i|t_{i-1})$, are estimated from a small section (970,995 tokens,47,831 distinct words) of the British National Corpus (BNC), which is a 100 million-word collection of British English, both written and spoken, developed by Oxford University Press (Burnard, 1995). The BNC was chosen for training the model because it is a POS-annotated corpus, which allows supervised training. In the implementation we use log probabilities to avoid underflow, and we report log probabilities in the sequel.

2.1 Hypotheses

If the HSPM is affected by frequency information, we can assume that it will be easier to process events with higher frequency or probability compared to those with lower frequency or probability. Under this general assumption, the overall difficulty of a sentence is expected to be measured or predicted by the mean size of probability decrease. That is, probability will drop faster in garden-path sentences than in control sentences (e.g. unambiguous sentences or ambiguous but non-garden-path sentences).

More importantly, the probability decrease pattern at disambiguating regions will predict the trends in the reading time data. All other things being equal, we might expect a reading time penalty for a garden-path region when the size of the probability decrease at the disambiguating region of a garden-path sentence will be greater than that of control sentences. This is a simple and intuitive assumption that can be easily tested. We could have formed the sum over all possible POS sequences in association with the word strings, but for the present study we simply used the Viterbi path: justifying this because this is the best single-path approximation to the joint probability.

Lastly, re-ranking of POS sequences is expected to predict reanalysis of lexical categories. This is because re-ranking in the tagger is parallel to reanalysis in human subjects, which is known to be cognitively costly.

2.2 Materials

In this study, five different types of ambiguity were tested including Lexical Category ambiguity, Reduced-relative ambiguity (RR ambiguity), Preposition-phrase attachment ambiguity (PP ambiguity), Direct-object/Sentential-complement ambiguity (DO/SC ambiguity), and Clausal Boundary ambiguity. The following are example sentences for each ambiguity type, shown with the ambiguous region italicized and the disambiguating region bolded. All of the example sentences are garden-path sentneces.

(3) Lexical Category ambiguity
The foreman knows that the warehouse *prices* **the** beer very modestly.

(4) RR ambiguity
The horse *raced* past the barn **fell**.

(5) PP ambiguity
Katie laid the dress *on the floor* **onto** the bed.

(6) DO/SC ambiguity
He forgot Pam **needed** a ride with him.

(7) Clausal Boundary ambiguity
Though George kept on reading *the story* really **bothered** him.

The test materials are constructed such that a garden-path sentence and its control sentence share exactly the same word sequence except for the disambiguating word so that extraneous variables such as word frequency effect can be controlled. We inherit this careful design.

In this study, a total of 76 sentences were tested: 10 for lexical category ambiguity, 12 for RR ambiguity, 20 for PP attachment ambiguity, 16 for DO/SC ambiguity, and 18 for clausal boundary ambiguity. This set of materials is, to our knowledge, the most comprehensive yet subjected to this type of study. The sentences are directly adopted from various psycholinguistic studies (Frazier, 1978; Trueswell, 1996; Ferreira and Henderson, 1986).

As a baseline test case of the tagger, the well-established asymmetry between subject- and object-relative clauses was tested as shown in (8).

(8) a. The editor who kicked the writer fired the entire staff. (Subject-relative)
b. The editor who the writer kicked fired the entire staff. (Object-relative)

The reading time advantage of subject-relative clauses over object-relative clauses is robust in English (Traxler et al., 2002) as well as other languages (Mak et al., 2002; Homes et al., 1981). For this test, materials from Traxler et al. (2002) (96 sentences) are used.

3 Results

3.1 The Probability Decrease per Word

Unambiguous sentences are usually longer than garden-path sentences. To compare sentences of different lengths, the joint probability of the whole sentence and tags was divided by the number of words in the sentence. The result showed that the average probability decrease was greater in garden-path sentences compared to their unambiguous control sentences. This indicates that garden-path sentences are more difficult than unambiguous sentences, which is consistent with empirical findings.

Probability decreased faster in object-relative sentences than in subject relatives as predicted. In the psycholinguistics literature, the comparative difficulty of object-relative clauses has been explained in terms of verbal working memory (King and Just, 1991), distance between the gap and the filler (Bever and McElree, 1988), or perspective shifting (MacWhinney, 1982). However, the test results in this study provide a simpler account for the effect. That is, the comparative difficulty of an object-relative clause might be attributed to its less frequent POS sequence. This account is particularly convincing since each pair of sentences in the experiment share the exactly same set of words except their order.

3.2 Probability Decrease at the Disambiguating Region

A total of 30 pairs of a garden-path sentence and its ambiguous, non-garden-path control were tested for a comparison of the probability decrease at the disambiguating region. In 80% of the cases, the probability drops more sharply in garden-path sentences than in control sentences at the critical word. The test results are presented in (9) with the number of test sets for each ambiguous type and the number of cases where the model correctly predicted reading-time penalty of garden-path sentences.

(9) Ambiguity Type (Correct Predictions/Test Sets)
a. Lexical Category Ambiguity (4/4)
b. PP Attachment Ambiguity (10/10)
c. RR Ambiguity (3/4)
d. DO/SC Ambiguity (4/6)
e. Clausal Boundary Ambiguity (3/6)

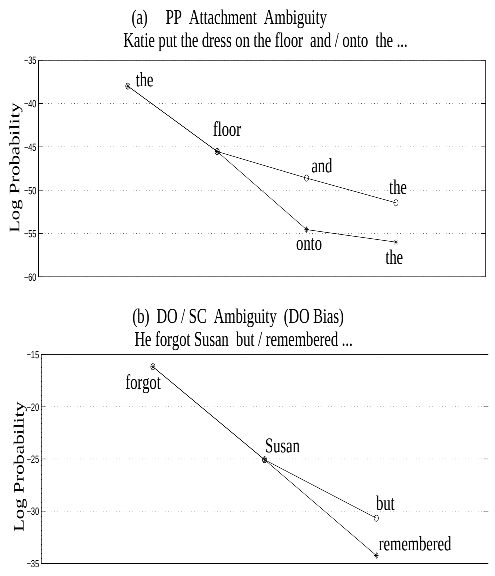


Figure 1: Probability Transition (Garden-Path vs. Non Garden-Path)

(a) $-\circ-$: Non-Garden-Path (Adjunct PP), $-*-$: Garden-Path (Complement PP)
(b) $-\circ-$: Non-Garden-Path (DO-Biased, DO-Resolved), $-*-$: Garden-Path (DO-Biased, SC-Resolved)

The two graphs in Figure 1 illustrate the comparison of probability decrease between a pair of sentence. The *y*-axis of both graphs in Figure 1 is log probability. The first graph compares the probability drop for PP ambiguity (*Katie put the dress on the floor and/onto the bed....*) The empirical result for this type of ambiguity shows that reading time penalty is observed when the second PP, *onto the bed*, is introduced, and there is no such effect for the other sentence. Indeed, the sharper probability drop indicates that the additional PP is less likely, which makes a prediction of a comparative processing difficulty. The second graph exhibits the probability comparison for the DO/SC ambiguity. The verb *forget* is a DO-biased verb and thus processing difficulty is observed when it has a sentential complement. Again, this effect was replicated here.

The results showed that the disambiguating word given the previous context is more difficult in garden-path sentences compared to control sentences. There are two possible explanations for the processing difficulty. One is that the POS sequence of a garden-path sentence is less probable than that of its control sentence. The other account is that the disambiguating word in a garden-path sentence is a lower frequency word compared to that of its control sentence.

For example, slower reading time was observed in (10a) and (11a) compared to (10b) and (11b) at the disambiguating region that is bolded.

(10) Different POS at the Disambiguating Region

a. Katie laid the dress *on the floor* **onto** (-57.80) the bed.

b. Katie laid the dress *on the floor* **after** (-55.77) her mother yelled at her.

(11) Same POS at the Disambiguating Region

a. The umpire helped the child ***on*** (-42.77) third base.

b. The umpire helped the child ***to*** (-42.23) third base.

The log probability for each disambiguating word is given at the end of each sentence. As expected, the probability at the disambiguating region in (10a) and (11a) is lower than in (10b) and (11b) respectively. The disambiguating words in (10) have different POS's; Preposition in (10a) and Conjunction (10b). This suggests that the probabilities of different POS sequences can account for different reading time at the region. In (11), however, both disambiguating words are the same POS (i.e. Preposition) and the POS sequences for both sentences are identical. Instead, "on" and "to", have different frequencies and this information is reflected in the conditional probability $P(word_i|state)$. Therefore, the slower reading time in (11b) might be attributable to the lower frequency of the disambiguating word, "to" compared to "on".

3.3 Probability Re-ranking

The probability re-ranking reported in Corley and Crocker (2000) was replicated. The tagger successfully resolved the ambiguity by reanalysis when the ambiguous word was immediately followed by the disambiguating word (e.g. Without *her* **he** was lost.). If the disambiguating word did not immediately follow the ambiguous region, (e.g. Without *her* contributions **would** be very inadequate.) the ambiguity is sometimes incorrectly resolved.

When revision occurred, probability dropped more sharply at the revision point and at the disambiguation region compared to the control sentences. When the ambiguity was not correctly resolved, the probability comparison correctly modeled the comparative difficulty of the garden-path sentences

Of particular interest in this study is RR ambiguity resolution. The tagger predicted the processing difficulty of the RR ambiguity with probability re-ranking. That is, the tagger initially favors the main-verb interpretation for the ambiguous *-ed* form, and later it makes a repair when the ambiguity is resolved as a past-participle.

The RR ambiguity is often categorized as a syntactic ambiguity, but the results suggest that the ambiguity can be resolved locally and its processing difficulty can be detected by a finite state model. This suggests that we should be cautious in assuming that a structural explanation is needed for the RR ambiguity resolution, and it could be that similar cautions are in order for other ambiguities usually seen as syntactic.

4 Discussion

The current study explores Corley and Crocker's model(2000) further on the model's account of human sentence processing data seen in empirical studies. Although there have been studies on a POS tagger evaluating it as a potential cognitive module of lexical category disambiguation, there has been little work that tests it as a modeling tool of syntactically ambiguous sentence processing.

The findings here suggest that a statistical POS tagging system is more informative than Crocker and Corley demonstrated. It has a predictive power of processing delay not only for lexically ambiguous sentences but also for structurally garden-pathed sentences. This model is attractive since it is computationally simpler and requires few statistical parameters. More importantly, it is clearly defined what predictions can be and cannot be made by this model. This allows systematic testability and refutability of the model unlike some other probabilistic frameworks. Also, the model training and testing is transparent and observable, and true probability rather than transformed weights are used, all of which makes it easy to understand the mechanism of the proposed model.

Although the model we used in the current study is not a novelty, the current work largely differs from the previous study in its scope of data used and the interpretation of the model for human sentence processing. Corley and Crocker clearly state that their model is strictly limited to lexical ambiguity resolution, and their test of the model was bounded to the noun-verb ambiguity. However, the findings in the current study play out differently. The experiments conducted in this study are parallel to empirical studies with regard to the design of experimental method and the test material. The garden-path sentences used in this study are authentic, most of them are selected from the cited literature, not conveniently coined by the authors. The word-by-word probability comparison between garden-path sentences and their controls is parallel to the experimental design widely adopted in empirical studies in the form of region-by-region reading or eye-gaze time comparison. In the word-by-word probability comparison, the model is tested whether or not it correctly predicts the comparative processing difficulty at the garden-path region. Contrary to the major claim made in previous empirical studies, which is that the garden-path phenomena are either modeled by syntactic principles or by structural frequency, the findings here show that the same phenomena can be predicted without such structural information.

Therefore, the work is neither a mere extended application of Corley and Crocker's work to a broader range of data, nor does it simply confirm earlier observations that finite state machines might accurately account for psycholinguistic results to some degree. The current study provides more concrete answers to what finite state machine is relevant to what kinds of processing difficulty and to what extent.

5 Conclusion

Our studies show that, at least for the sample of test materials that we culled from the standard literature, a statistical POS tagging system can predict processing difficulty in structurally ambiguous garden-path sentences. The statistical POS tagger was surprisingly effective in modeling sentence processing data, given the locality of the probability distribution. The findings in this study provide an alternative account for the garden-path effect observed in empirical studies, specifically, that the slower processing times associated with garden-path sentences are due in part to their relatively unlikely POS sequences in comparison with those of non-garden-path sentences and in part to differences in the emission probabilities that the tagger learns. One attractive future direction is to carry out simulations that compare the evolution of probabilities in the tagger with that in a theoretically more powerful model trained on the same data, such as an incremental statistical parser (Wang et al., 2004; Roark, 2001). In so doing we can find the places where the prediction problem faced both by the HSPM and the machines that aspire to emulate it actually warrants the greater power of structurally sensitive models, using this knowledge to mine large corpora for future experiments with human subjects.

We have not necessarily cast doubt on the hypothesis that the HSPM makes crucial use of structural information, but we have demonstrated that much of the relevant behavior can be captured in a simple model. The 'structural' regularities that we observe are reasonably well encoded into this model. For purposes of initial real-time processing it could be that the HSPM is using a similar encoding of structural regularities into convenient probabilistic or neural form. It is as yet unclear what the final form of a cognitively accurate model along these lines would be, but it is clear from our study that it is worthwhile, for the sake of clarity and explicit testability, to consider models that are simpler and more precisely specified than those assumed by dominant theories of human sentence processing.

Acknowledgments

This project was supported by the Cognitive Science Summer 2004 Research Award at the Ohio State University. We acknowledge support from NSF grant IIS 0347799.

References

T. G. Bever and B. McElree. Empty categories access their antecedents during comprehension. *Linguistic Inquiry*, 19:35–43, 1988.

L Burnard. *Users Guide for the British National Corpus*. British National Corpus Consortium, Oxford University Computing Service, 1995.

S. Corley and M. W Crocker. *The Modular Statistical Hypothesis: Exploring Lexical Category Ambiguity*. Architectures and Mechanisms for Language Processing, M. Crocker, M. Pickering. and C. Charles (Eds.) Cambridge University Press, 2000.

F. Ferreira and J. Henderson. Use of verb information in syntactic parsing: Evidence from eye movements and word-by-word self-paced reading. *Journal of Experimental Psychology*, 16: 555–568, 1986.

L. Frazier. On comprehending sentences: Syntactic parsing strategies. *Ph.D. dissertation, University of Massachusetts*, Amherst, MA, 1978.

L. Frazier and K. Rayner. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14: 178–210, 1982.

V. M. Homes, J. O'Regan, and K.G. Evensen. Eye fixation patterns during the reading of relative clause sentences. *Journal of Verbal Learning and Verbal Behavior*, 20:417–430, 1981.

J. King and M. A. Just. Individual differences in syntactic processing: The role of working memory. *Journal of Memory and Language*, 30:580–602, 1991.

B. MacWhinney. Basic syntactic processes. *Language acquisition; Syntax and semantics, S. Kuczaj (Ed.)*, 1:73–136, 1982.

W. M. Mak, Vonk W., and H. Schriefers. The influence of animacy on relative clause processing. *Journal of Memory and Language,*, 47:50–68, 2002.

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

B. Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27 (2):249–276, 2001.

M. J. Traxler, R. K. Morris, and R. E. Seely. Processing subject and object relative clauses: evidence from eye movements. *Journal of Memory and Language*, 47:69–90, 2002.

J. C. Trueswell. The role of lexical frequency in syntactic ambiguity resolution. *Journal of Memory and Language*, 35:556–585, 1996.

A. Viterbi. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Transactions of Information Theory*, 13: 260–269, 1967.

W. Wang, A. Stolcke, and M. P. Harper. The use of a linguistically motivated language model in conversational speech recognition. In *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing*, Montreal, Canada, 2004.

Semantic Discourse Segmentation and Labeling for Route Instructions

Nobuyuki Shimizu
Department of Computer Science
State University of New York at Albany
Albany, NY 12222, USA
nobuyuki@shimizu.name

Abstract

In order to build a simulated robot that accepts instructions in unconstrained natural language, a corpus of 427 route instructions was collected from human subjects in the office navigation domain. The instructions were segmented by the steps in the actual route and labeled with the action taken in each step. This flat formulation reduced the problem to an IE/Segmentation task, to which we applied Conditional Random Fields. We compared the performance of CRFs with a set of hand-written rules. The result showed that CRFs perform better with a 73.7% success rate.

1 Introduction

To have seamless interactions with computers, advances in task-oriented deep semantic understanding are of utmost importance. The examples include tutoring, dialogue systems and the one described in this paper, a natural language interface to mobile robots. Compared to more typical text processing tasks on newspapers for which we attempt shallow understandings and broad coverage, for these domains vocabulary is limited and very strong domain knowledge is available. Despite this, deeper understanding of unrestricted natural language instructions poses a real challenge, due to the incredibly rich structures and creative expressions that people use. For example,

> ”Just head straight through the hallway ignoring the rooms to the left and right of you, but while going straight your going to eventually see a room facing you, which is north, enter it.”

> ”Head straight. continue straight past the first three doors until you hit a corner. On that corner there are two doors, one straight ahead of you and one on the right. Turn right and enter the room to the right and stop within.”

These utterances are taken from an office navigation corpus collected from undergrad volunteers at SUNY/Albany. There is a good deal of variety.

Previous efforts in this domain include the classic SHRDLU program by Winograd (1972), using a simulated robot, and the more ambitious IBL (Instruction-based Learning for Mobile Robots) project (Lauria et al, 2001) which tried to integrate vision, voice recognition, natural language understanding and robotics. This group has yet to publish performance statistics. In this paper we will focus on the application of machine learning to the understanding of written route instructions, and on testing by following the instructions in a simulated office environment.

2 Task

2.1 Input and Output

Three inputs are required for the task:

- Directions for reaching an office, written in unrestricted English.
- A description of the building we are traveling through.
- The agent’s initial position and orientation.

The output is the location of the office the directions aim to reach.

2.2 Corpus Collection

In an experiment to collect the corpus, (Haas, 1995) created a simulated office building modeled after the actual computer science department at SUNY/Albany. This environment was set up like a popular first person shooter game such as Doom, and the subject saw a demonstration of the route he/she was asked to describe. The subject wrote directions and sent them to the experimenter, who sat at another computer in the next room. The experimenter tried to follow the directions; if he reaches the right destination, the subject got $1. This process took place 10 times for each subject; instructions that the experimenter could not follow correctly were not added to the corpus. In this manner, they were able to elicit 427 route instructions from the subject pool of 44 undergraduate students.

2.3 Abstract Map

To simplify the learning task, the map of our computer science department was abstracted to a graph. Imagine a track running down the halls of the virtual building, with branches into the office doors. The nodes of the graph are the intersections, the edges are the pieces of track between them. We assume this map can either be prepared ahead of time, or dynamically created as a result of solving Simultaneous Localization and Mapping (SLAM) problem in robotics (Montemerlo et al, 2003).

2.4 System Components

Since it is difficult to jump ahead and learn the whole input-output association as described in the task section, we will break down the system into two components.

Front End:
$RouteInstruction \rightarrow ActionList$
Back End:
$ActionList \times Map \times Start \rightarrow Goal$

The front-end is an information extraction system, where the system extracts how one should move from a route instruction. The back-end is a reasoning system which takes a sequence of moves and finds the destination in the map. We will first describe the front-end, and then show how to integrate the back-end to it.

One possibility is to keep the semantic representation close to the surface structure, including under-specification and ambiguity, and leaving the back-end to resolve the ambiguity. We will pursue a different route. The disambiguation will be done in the front-end; the representation that it passes to the back-end will be unambiguous, describing at most one path through the building. The task of the back-end is simply to check the sequence of moves the front-end produced against the map and see if there is a path leading to a point in the map or not. The reason for this is two fold. One is to have a minimal annotation scheme for the corpus, and the other is to enable the learning of the whole task including the disambiguation as an IE problem.

3 Semantic Analysis

Note that in this paper, given an instruction, one *step* in the instruction corresponds to one *action* shown to the subject, one *episode* of action detection and tracking, and one *segment* of the text.

In order to annotate unambiguously, we need to detect and track both landmarks and actions. A **landmark** is a hallway or a door, and an **action** is a sequence of a few moves one will make with respect to a specific landmark.

The moves one can make in this map are:
(M1). Advancing to x,
(M2). Turning left/right to face x, and
(M3). Entering x.

Here, x is a landmark. Note that all three moves have to do with the same landmark, and two or three moves on the same landmark constitute one action. An action is ambiguous until x is filled with an unambiguous landmark. The following is a made-up example in which each move in an action is mentioned explicitly.

a. ”Go down the hallway to the second door on the right. Turn right. Enter the door.”

But you could break it down even further.

b. ”Go down the hallway. You will see two doors on the right. Turn right and enter the second.”

One can add any amount of extra information to an instruction and make it longer, which people seem to do. However, we see the following as well.

c. ”Enter the second door on the right.”

In one sentence, this sample contains the advance, the turn and the entering. In the corpus, the norm is to assume the move (M1) when an expression indicating the move (M2) is present. Similarly, an expression of move (M3) often implicitly assumes the move (M1) and (M2). However, in some cases they are explicitly stated, and when this happens, the action that involves the same landmark must be tracked across the sentences.

Since all three samples result in the same action, for the back-end it is best not to differentiate the three. In order to do this, actions must be tracked just like landmarks in the corpus.

The following two samples illustrate the need to track actions.

d. ”Go down the hallway until you see two doors. Turn right and enter the second door on the right.”

In this case, there is only one action in the instruction, and ”turn right” belongs to the action *”advance to the second door on the right, and then turn right to face it, and then enter it.”*

e. ”Proceed to the first hallway on the right. Turn right and enter the second door on the right.”

There are two actions in this instruction. The first is *”advance to the first hallway on the right, and then turn right to face the hallway.”* The phrase ”turn right” belongs to this first action. The second action is the same as the one in the example (d). Unless we can differentiate between the two, the execution of the unnecessary turn results in failure when following the instructions in the case (d).

This illustrates the need to track actions across a few sentences. In the last example, it is important to realize that ”turn right” has something to do with a door, so that it means ”turn right to face a door”. Furthermore, since ”enter the second door on the right” contains ”turning right to face a door” in its semantics as well, they can be thought of as the same action. Thus, the critical feature required in the annotation scheme is to track actions and landmarks.

The simplest annotation scheme that can show how actions are tracked across the sentences is to segment the instruction into different episodes of action detection and tracking. Note that each episode corresponds to exactly one action shown to the subject during the experiment. The annotation is based on the semantics, not on the the *mentions* of moves or landmarks. Since each segment involves exactly one landmark, we can label the segment with an action and a specific landmark. For example,

Token	Node Part	Transition Part
make	$\langle$*B-GHL1*$, 0\rangle$	$\langle$*B-GHL1*, *I-GHL1*$, 0, 1\rangle$
left	$\langle$*I-GHL1*$, 1\rangle$	$\langle$*I-GHL1*, *I-GHL1*$, 1, 2\rangle$
,	$\langle$*I-GHL1*$, 2\rangle$	$\langle$*I-GHL1*, *B-EDR1*$, 2, 3\rangle$
first	$\langle$*B-EDR1*$, 3\rangle$	$\langle$*B-EDR1*, *I-EDR1*$, 3, 4\rangle$
door	$\langle$*I-EDR1*$, 4\rangle$	$\langle$*I-EDR1*, *I-EDR1*$, 4, 5\rangle$
on	$\langle$*I-EDR1*$, 5\rangle$	$\langle$*I-EDR1*, *I-EDR1*$, 5, 6\rangle$
the	$\langle$*I-EDR1*$, 6\rangle$	$\langle$*I-EDR1*, *I-EDR1*$, 6, 7\rangle$
right	$\langle$*I-EDR1*$, 7\rangle$	

Table 1: Example Parts: linear-chain CRFs

GHR1 := ”advance to the first hallway on the right, then turn right to face it.”

EDR2 := ”advance to the second door on the right, then turn right to face it, then enter it.”

GHLZ := ”advance to the hallway on the left at the end of the hallway, then turn left to face it.”

EDSZ := ”advance to the door straight ahead of you, then enter it.”

Note that GH=go-hall, ED=enter-door, R1=first-right, LZ=left-at-end, SZ=ahead-of-you. The total number of possible actions is 15.

This way, we can reduce the front-end task into a sequence of tagging tasks, much like the noun phrase chunking in the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). Given a sequence of input tokens that forms a route instruction, a sequence of output labels, with each label matching an input token was prepared. We annotated with the BIO tagging scheme used in syntactic chunkers (Ramshaw and Marcus, 1995).

```
make        B-GHL1
left        I-GHL1
,           I-GHL1
first       B-EDR1
door        I-EDR1
on          I-EDR1
the         I-EDR1
right       I-EDR1
```

4 Systems

4.1 System 1: CRFs

4.1.1 Model: A Linear-Chain Undirected Graphical Model

From the output labels, we create the parts in a linear-chain undirected graph (Table 1). Our use of term **part** is based on (Bartlett et al, 2004).

For each pair (x^i, y^i) in the training set, x^i is the token (in the first column, Table 1), and y^i

Transition $\langle L0, L, j-1, j\rangle$	Node $\langle L, j\rangle$
no lexicalization	no lexicalization
	x_{j-4}
	x_{j-3}
	x_{j-2}
	x_{j-1}
	x_j
	x_{j+1}
	x_{j+2}
	x_{j+3}
	x_{j-1}, x_j
	x_{j+0}, x_{j+1}

Table 2: Features

is the part (in the second and third column, Table 1). There are two kinds of parts: node and transition. A node part tells us the position and the label, $\langle$*B-GHL1*$, 0\rangle$, $\langle$*I-GHL1*$, 1\rangle$, and so on. A transition part encodes a transition. For example, between tokens 0 and 1 there is a transition from tag B-GHL1 to I-GHL1. The part that describes this transition is: $\langle$*B-GHL1*, *I-GHL1*$, 0, 1\rangle$.

We factor the score of this linear node-transition structure as the sum of the scores of all the parts in y, where the score of a part is again the sum of the feature weights for that part.

To score a pair (x^i, y^i) in the training set, we take each part in y^i and check the features associated with it via lexicalization. For example, a part $\langle$*I-GHL1*$, 1\rangle$ could give rise to binary features such as,

- Does (x^i, y^i) contain a label ”I-GHL1”? (No Lexicalization)
- Does (x^i, y^i) contain a token ”left” labeled with ”I-GHL1”? (Lexicalized by x_1)
- Does (x^i, y^i) contain a token ”left” labeled with ”I-GHL1” that's preceded by ”make”? (Lexicalized by x_0, x_1)

and so on. The features used in this experiment are listed in Table 2.

If a feature is present, the feature weight is added. The sum of the weights of all the parts is the score of the pair (x^i, y^i). To represent this summation, we write $s(x^i, y^i) = \mathbf{w}^\top \mathbf{f}(x^i, y^i)$ where $\mathbf{f}$ represents the feature vector and $\mathbf{w}$ is the weight vector. We could also have $\mathbf{w}^\top \mathbf{f}(x^i, \{p\})$ where p is a single part, in which case we just write $s(p)$.

Assuming an appropriate feature representation as well as a weight vector $\mathbf{w}$, we would like to find the highest scoring $y = argmax_{y'}(\mathbf{w}_k^\top \mathbf{f}(y', x))$ given an input sequence x. We next present a version of this decoding algorithm that returns the best y consistent with the map.

4.1.2 Decoding: the Viterbi Algorithm and Inferring the Path in the Map

The action labels are unambiguous; given the current position, the map, and the action label, there is only one position one can go to. This backend computation can be integrated into the Viterbi algorithm. The function 'go' takes a pair of (action label, start position) and returns the end position or null if the action cannot be executed at the start position according to the map. The algorithm chooses the best among the label sequences with a legal path in the map, as required by the condition $(cost > bestc \wedge end \neq null)$. Once the model is trained, we can then use the modified version of the Viterbi algorithm (Algorithm 4.1) to find the destination in the map.

```
Algorithm 4.1: DECODE PATH($x, n, start, go$)

for each label $y_1$
  $node[0][y_1].cost \leftarrow s(\langle y_1, 0\rangle)$
  $node[0][y_1].end \leftarrow start;$
for $j \leftarrow 1$ to $n-1$
  for each label $y_{j+1}$
    $bestc \leftarrow -\infty;$
    $end \leftarrow null;$
    for each label $y_j$
      $cost \leftarrow node[j][y_j].cost$
          $+s(\langle y_j, y_{j+1}, j, j+1\rangle)$
          $+s(\langle y_{j+1}, j+1\rangle);$
      $end \leftarrow node[j][y_j].end;$
      if $(y_j \neq y_{j+1})$
        $end \leftarrow go(y_{j+1}, end);$
      if $(cost > bestc \wedge end \neq null)$
        $bestc \leftarrow cost;$
    if $(bestc \neq -\infty)$
      $node[j+1][y_{j+1}].cost \leftarrow bestc;$
      $node[j+1][y_{j+1}].end \leftarrow end;$
$bestc \leftarrow -\infty;$
$end \leftarrow null;$
for each label $y_n$
  if $(node[j][y_n].cost > bestc)$
    $bestc \leftarrow node[j][y_n].cost;$
    $end \leftarrow node[j][y_n].end;$
return $(bestc, end)$
```

4.1.3 Learning: Conditional Random Fields

Given the above problem formulation, we trained the linear-chain undirected graphical model as Conditional Random Fields (Lafferty et al, 2001; Sha and Pereira, 2003), one of the best performing chunkers. We assume the probability of seeing y given x is

$$P(y|x) = \frac{exp(s(x,y))}{\sum_{y'} exp(s(x,y'))}$$

where y' is all possible labeling for x , Now, given a training set $T = \{(x^i y^i)\}_{i=1}^m$, We can learn the weights by maximizing the log-likelihood, $\sum_i logP(y^i|x^i)$. A detailed description of CRFs can be found in (Lafferty et al, 2001; Sha and Pereira, 2003; Malouf, 2002; Peng and McCallum, 2004). We used an implementation called CRF++ which can be found in (Kudo, 2005)

4.2 System 2: Baseline

Suppose we have clean data and there is no need to track an action across sentences or phrases. Then, the properties of an action are mentioned exactly once for each episode.

For example, in *"go straight and make the first left you can, then go into the first door on the right side and stop"* , LEFT and FIRST occur exactly once for the first action, and FIRST, DOOR and RIGHT are found exactly once in the next action. In a case like that, the following baseline algorithm should work well.

- Find all the mentions of LEFT/RIGHT,
- For each occurrence of LEFT/RIGHT, look for an ordinal number, LAST, or END (= end of the hallway) nearby,
- Also, for each LEFT/RIGHT, look for a mention of DOOR. If DOOR is mentioned, the action is about entering a door.
- If DOOR is not mentioned around LEFT/RIGHT, then the action is about going to a hallway by default,
- If DOOR is mentioned at the end of an instruction without LEFT/RIGHT, then the action is to go straight into the room.
- Put the sequence of action labels together according to the mentions collected.

	count	average length
GHL1	128	8.5
GHL2	4	7.7
GHLZ	36	14.4
GHR1	175	10.8
GHR2	5	15.8
GHRZ	42	13.6
EDL1	98	10.5
EDL2	81	12.3
EDL3	24	13.9
EDLZ	28	13.7
EDR1	69	10.4
EDR2	55	12.9
EDR3	6	13.0
EDRZ	11	16.4
EDSZ	55	16.2

Table 3: Steps found in the dataset

In this case, all that's required is a dictionary of how a word maps to a concept such as DOOR. In this corpus, "door", "office", "room", "doorway" and their plural forms map to DOOR, and the ordinal number 1 will be represented by "first" and "1st", and so on.

5 Dataset

As noted, we have 427 route instructions, and the average number of steps was 1.86 steps per instruction. We had 189 cases in which a sentence boundary was found in the middle of a step. Table 3 shows how often action steps occurred in the corpus and average length of the segments.

One thing we noticed is that somehow people do not use a short phrase to say the equivalent of "enter the door straight ahead of you", as seen by the average length of EDSZ. Also, it is more common to say the equivalent of "take a right at the end of the hallway" than that of "go to the second hallway on the right", as seen by the count of GHR2 and GHRZ. The distribution is highly skewed; there are a lot more GHL1 than GHL2.

6 Results

We evaluated the performance of the systems using three measures: overlap match, exact match, and instruction follow through, using 6-fold crossvaliadation on 427 samples. Only the action chunks were considered for exact match and overlap match. Overlap match is a lenient measure that considers a segmentation or labeling to be cor-

Exact Match	Recall	Precision	F-1
CRFs	66.0%	67.0%	66.5%

Overlap Match	Recall	Precision	F-1
Baseline	62.8%	49.9%	55.6%
CRFs	85.7%	87.0%	86.3%

Instruction Follow Through	success rate
Baseline	39.5%
CRFs	73.7%

Table 4: Recall, Precision, F-1 and Success Rate

rect if it overlaps with any of the annotated labels. Instruction follow through is the success rate for reaching the destination, and the most important measure of the performance in this domain. Since the baseline algorithm does not identify the token labeled with B-prefix, no exact match comparison is made. The result (Table 4) shows that CRFs perform better with a 73.7% success rate.

7 Future Work

More complex models capable of representing landmarks and actions separately may be applicable to this domain, and it remains to be seen if such models will perform better. Also, some form of co-reference resolution or more sophisticated action tracking should also be considered.

Acknowledgement

We thank Dr. Andrew Haas for introducing us to the problem, collecting the corpus and being very supportive in general.

References

P. Bartlett, M. Collins, B. Taskar and D. McAllester. 2004. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems (NIPS)*

A. Haas 1995. Testing a Simulated Robot that Follows Directions. *unpublished*

T. Kudo 2005. CRF++: Yet Another CRF toolkit. Available at *http://chasen.org/˜taku/software/CRF++/*

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of International Conference on Machine Learning* .

R. Malouf. 2002. A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In *Proceedings of Conference of Computational Natural Language Learning*

F. Peng and A. McCallum. 2004. Accurate Information Extraction from Research Papers using Conditional Random Fields. In *Proceedings of Human Language Technology Conference* .

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference* .

S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, and E. Klein. 2001. Personal Robot Training via Natural-Language Instructions. *IEEE Intelligent Systems*, 16:3, pp. 38-45.

C. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing.* MIT Press.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. 2003. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

L. Ramshaw and M. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of Third Workshop on Very Large Corpora. ACL*

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of Conference of Computational Natural Language Learning* .

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

Investigations on Event-Based Summarization

Mingli Wu
Department of Computing
The Hong Kong Polytechnic University
Kowloon, Hong Kong
csmlwu@comp.polyu.edu.hk

Abstract

We investigate independent and relevant event-based extractive mutli-document summarization approaches. In this paper, events are defined as event terms and associated event elements. With independent approach, we identify important contents by frequency of events. With relevant approach, we identify important contents by PageRank algorithm on the event map constructed from documents. Experimental results are encouraging.

1 Introduction

With the growing of online information, it is inefficient for a computer user to browse a great number of individual news documents. Automatic summarization is a powerful way to overcome such difficulty. However, the research literature demonstrates that machine summaries need to be improved further.

The previous research on text summarization can date back to (Luhn 1958) and (Edmundson 1969). In the following periods, some researchers focus on extraction-based summarization, as it is effective and simple. Others try to generate abstractions, but these works are highly domain-dependent or just preliminary investigations. Recently, query-based summarization has received much attention. However, it is highly related to information retrieval, another research subject. In this paper, we focus on generic summarization. News reports are crucial to our daily life. In this paper, we focus on effective summarization approaches for news reports.

Extractive summarization is widely investigated in the past. It extracts part of document(s) based on some weighting scheme, in which different features are exploited, such as position in document, term frequency, and key phrases. Recent extraction approaches may also employ machine learning approaches to decide which sentences or phrases should be extracted. They achieve preliminary success in different application and wait to be improved further.

Previous extractive approaches identify the important content mainly based on terms. Bag of words is not a good representation to specify an event. There are multiple possible explanations for the same collection of words. A predefined template is a better choice to represent the event. However it is domain-dependent and need much effort to create and fill it. This tension motivates us to seek a balance between effective implementation and deep understanding.

According to related works (Filatovia and Hatzivassiloglou, 2004) (Vanderwende et al., 2004), we assume that event may be a natural unit to convey meanings of documents. In this paper, event is defined as the collection of event terms and associated event elements in clause level. Event terms express the meaning of actions themselves, such as “incorporate”. In addition to verbs, action nouns can also express meaning of actions and should be regarded as event terms. For example, “incorporation” is action noun. Event elements include named entities, such as person name, organization name, location, time. These named entities are tagged with GATE (Cunningham et al., 2002). Based on our event definition, independent and relevant event-based approaches are investigated in this research. Experiments show that both of them achieve encouraging results.

The related works are discussed in Section 2. Independent event-based summarization approach is described in Section 3. Relevant event-based summarization approach is described in Section 4. Section 5 presents the experiments and evaluations. Then the strength and limitation of our approaches are discussed in Section 6. Finally, we conclude the work in Section 7.

2 Related Work

Term-based extractive summarization can date back to (Luhn, 1958) and (Edmundson, 1969). This approach is simple but rather applicable. It represents the content of documents mainly by bag of words. Luhn (1958) establishes a set of "significant" words, whose frequency is between a higher bound and a lower bound. Edmundson (1969) collects common words, cue words, title/heading words from documents. Weight scores of sentences are computed based on type/frequency of terms. Sentences with higher scores will be included in summaries. Later researchers adopt tf*idf score to discriminate words (Brandow et al., 1995) (Radev et al., 2004). Other surface features are also exploited to extract important sentence, such as position of sentence and length of sentence (Teufel and Moens, 1999) (Radev et al., 2004). To make the extraction model suitable for documents in different domains, recently machine learning approaches are widely employed (Kupiec et al., 1995) (Conroy and Schlesinger, 2004).

To represent deep meaning of documents, other researchers have investigated different structures. Barzilay and Elhadad (1997) segment the original text and construct lexical chains. They employ strong chains to represent important parts of documents. Marcu (1997) describes a rhetorical parsing approach which takes unrestricted text as input and derives the rhetorical structure tree. They express documents with structure trees. Dejong (1978) adopts predefined templates to express documents. For each topic, the user predefines frames of expected information types, together with recognition criteria. However, these approaches just achieve moderate results.

Recently, event receives attention to represent documents. Filatovia and Hatzivassiloglou (2004) define event as action (verbs/action nouns) and named entities. After identifying actions and event entities, they adopt frequency weighting scheme to identify important sentence. Vanderwende et al. (2004) represent event by dependency triples. After analysis of triples they connect nodes (words or phrases) by way of semantic relationships. Yoshioka and Haraguchi (2004) adopt a similar approach to build a map, but they regard sentence as the nodes of the map. After construction of a map representation for documents, Vanderwende et al. (2004), and Yoshioka and Haraguchi (2004) all employ PageRank algorithm to select the important sentences. Although these approaches employ event representation and PageRank algorithm, it should be noted that our event representation is different with theirs. Our event representation is based on named entities and event terms, without help of dependency parsing. These previous event-based approaches achieved promising results.

3 Independent Event-based Summarization

Based on our observation, we assume that events in the documents may have different importance. Important event terms will be repeated and always occur with more event elements, because reporters hope to state them clearly. At the same time, people may omit time or location of an important event after they describe the event previously. Therefore in our research, event terms occurs in different circumstances will be assigned different weights. Event terms occur between two event elements should be more important than event terms occurring just beside one event elements. Event terms co-occurring with participants may be more important than event terms just beside time or location.

The approach on independent event-based summarization involves following steps.

1. Given a cluster of documents, analyze each sentence one at a time. Ignore sentences that do not contain any event element.

2. Tag the event terms in the sentence, which is between two event elements or near an event element with the distance limitation. For example, [Event Element A, <u>Even Term</u>, Event Element B], [<u>Event Term</u>, Event Element A], [Event Element A, <u>Event Term</u>]

3. Assign different weights to different event terms, according to contexts of event terms. Different weight configurations are described in Section 5.2. Contexts refer to number of event elements beside event terms and types of these event elements.

4. Get the average tf*idf score as the weight of every event term or event element. The algorithm is similar with Centroid.

5. Sum up the weights of event terms and event elements in a sentence.
6. Select the top sentences with highest weights, according to the length of summary.

4 Relevant Event-based Summarization

Independent event-based approaches do not exploit relevance between events. However, we think that it may be useful to identify important events. After a document is represented by events, relevant events are linked together. We made the assumption that important events may be mentioned often and events associated to important events may be important also. PageRank is a suitable algorithm to identify the importance of events from a map, according to the previous assumption. In the following sections, we will discuss how to represent documents by events and how to identify important event with PageRank algorithm.

4.1 Document Representation

We employ an event map to represent content of a document cluster, which is about a certain topic. In an event map, nodes are event terms or event elements, and edges represent association or modification between two nodes. Since the sentence is a natural unit to express meanings, we assume that all event terms in a sentence are all relevant and should be linked together. The links between every two nodes are undirectional.

In an ideal case, event elements should be linked to the associated event terms. At the same time, an event element may modify another element. For example, one element is a head noun and another one is the modifier. An event term (e.g., verb variants) may modify an event element or event term of another event. In this case, a full parser should be employed to get associations or modifications between different nodes in the map. Because the performance of current parsing technology is not perfect, an effective approach is to simulate the parse tree to avoid introducing errors of a parser. The simplifications are described as follows. Only event elements are attached with corresponding event terms. An event term will not be attached to an event element of another event. Also, an event element will not be attached to another event element. Heuristics are used to attach event elements with corresponding event terms.

Given a sentence "Andrew had become little more than a strong rainstorm early yesterday, moving across Mississippi state and heading for the north-eastern US", the event map is shown in Fig. 1. After each sentence is represented by a map, there will be multiple maps for a cluster of documents. If nodes from different maps are lexical match, they may denote same thing and should be linked. For example, if named entity "Andrew" occurred in Sentence A, B and C, then the three occurrences O_A, O_B and O_C will be linked as O_A—O_B, O_B—O_C, O_C—O_A. By this way, maps for sentences can be linked based on same concepts.

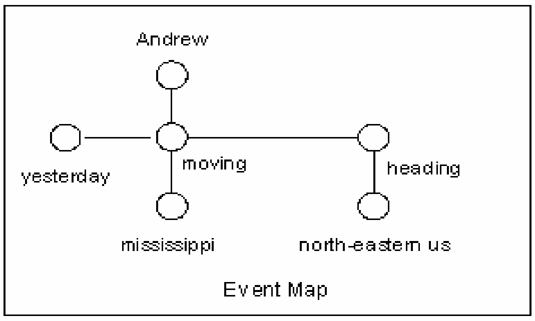


Figure 1. Document representation with event map

4.2 Importance Identification by PageRank

Given a whole map for a cluster of documents, the next step is to identify focus of these documents. Based on our assumption about important content in the previous section, PageRank algorithm (Page et al., 1998) is employed to fulfill this task. PageRank assumes that if a node is connected with more other nodes, it may be more likely to represent a salient concept. The nodes relevant to the significant nodes are closer to the salient concept than those not. The algorithm assigns the significance score to each node according to the number of nodes linking to it as well as the significance of the nodes. In PageRank algorithm, we use two directional links instead for every unidirectional link in Figure 1.

The equation to calculate the importance (indicated by *PR*) of a certain node *A* is shown as follows:

$$PR(A) = (1-d) + d\left(\frac{PR(B_1)}{C(B_1)} + \frac{PR(B_2)}{C(B_2)} + ... + \frac{PR(B_t)}{C(B_t)}\right)$$

Where B_1, B_2,…, B_t are all nodes which link to the node *A*. $C(B_i)$ is the number of outgoing links from the node B_i. The weight score of each node can be gotten by this equation recursively. *d* is the factor used to avoid the limitation of loop in the map structure. As the literature (Page et al., 1998) suggested, *d* is set as 0.85. The significance of each sentence to be included in the summary is then derived from the significance of the event terms and event elements it contains.

5 Evaluation

5.1 Dataset and Evaluation Metrics

DUC 2001 dataset is employed to evaluate our summarization approaches. It contains 30 clusters and a total of 308 documents. The number of documents in each cluster is between 3 and 20. These documents are from some English news agencies, such as Wall Street Journal. The contents of each cluster are about some specific topic, such as the hurricane in Florida. For each cluster, there are 3 different model summaries, which are provided manually. These model summaries are created by NIST assessors for the DUC task of generic summarization. Manual summaries with 50 words, 100 words, 200 words and 400 words are provided.

Since manual evaluation is time-consuming and may be subjective, the typical evaluation package, ROUGE (Lin and Hovy, 2003), is employed to test the quality of summaries. ROUGE compares the machine-generated summaries with manually provided summaries, based on unigram overlap, bigram overlap, and overlap with long distance. It is a recall-based measure and requires that the length of the summaries be limited to allow meaningful comparison. ROUGE is not a comprehensive evaluation method and intends to provide a rough description about the performance of machine generated summary.

5.2 Experimental Configuration

In the following experiments for independent event-based summarization, we investigate the effectiveness of the approach. In addition, we attempt to test the importance of contextual information in scoring event terms. The number of associated event terms and the type of event terms are considered to set the weights of event terms. The weights parameters in the following experiments are chosen according to empirical estimations.

Experiment 1: Weight of any entity is 1. Weight of any verb/action noun, which is between two entities or just beside one entity, is 1.

Experiment 2: Weight of any entity is 1. Weight of any verb/action noun, which is between two entities, is 3. Weight of any verb/action noun, which is just beside one entity, is 1.

Experiment 3: Weight of any entity is 1. Weight of any verb/action noun, which is between two entities and the first entity is person or organization, is 5. Weight of any verb/action noun, which is between two entities and the first entity is not person and not organization, is 3. Weight of any verb/action noun, which is just after a person or organization, is 2. Weight of any verb/action noun, which is just before one entity, is 1. Weight of any verb/action noun, which is just after one entity and the entity is not person and not organization, is 1.

In the following experiments, we investigate the effectiveness of our approaches on under different length limitation of summary. Based on the algorithm of experiment 3, we design experiment to generate summaries with length 50 words, 100 words, 200 words, 400 words. They are named **Experiment 4**, **Experiment 5**, **Experiment 3** and **Experiment 6**.

In other experiments for relevant event-based summarization, we investigate the function of relevance between events. The configurations are described as follows.

Experiment 7: Event terms and event elements are identified as we discussed in Section 3. In this experiment, event elements just include named entities. Occurrences of event terms or event elements are linked with by exact matches. Finally, the PageRank is employed to select important events and then important sentences.

Experiment 8: For reference, we select one of the four model summaries as the final summary for each cluster of documents. ROUGE is employed to evaluate the performance of these manual summaries.

5.3 Experimental Results

The experiment results on independent event-based summarization are shown in table 1. The results for relevant event-based summarization are shown in table 3.

	Exp. 1	Exp. 2	Exp. 3
Rouge-1	0.315	0.322	0.323
Rouge-2	0.049	0.055	0.055
Rouge-L	0.299	0.305	0.306

Table 1. Results on independent event-based summarization (summary with length of 200 words)

From table 1, we can see that results of Experiment 2 are better than those of Experiment 1. It proves our assumption that importance of event terms is different when these event terms occur with different number of event elements. Results of Experiment 3 are not significant better than those of Experiment 2, so it seems that the

assumption that importance of event terms is not very different when these event terms occur with different types of event elements. Another possible explanation is that after adjustment of the weight for event terms, the difference between the results of Experiment 2 and Experiment 3 may be extended.
\

	Exp. 4	Exp. 5	Exp. 3	Exp. 6
Rouge-1	0.197	0.249	0.323	0.382
Rouge-2	0.021	0.031	0.055	0.081
Rouge-L	0.176	0.231	0.306	0.367

Table 2. Results on independent event-based summarization (summary with different length)

Four experiments of table 2 show that performance of our event based summarization are getting better, when the length of summaries is expanded. One reason is that event based approach prefers sentences with more event terms and more event elements, so the preferred lengths of sentences are longer. While in a short summary, people always condense sentences from original documents, and use some new words to substitute original concepts in documents. Then the Rouge score, which evaluates recall aspect, is not good in our event-based approach. In contrast, if the summaries are longer, people will adopt detail event descriptions in original documents, and so our performance is improved.

	Exp. 7	Exp. 8
Rouge-1	0.325	0.595
Rouge-2	0.060	0.394
Rouge-L	0.305	0.586

Table 3. Results on relevant event-based summarization and a reference experiment (summary with length of 200 words)

In table 3, we found the Rouge-score of relevant event-based summarization (Experiment 7) is better than independent approach (Experiment 1). In Experiment 1, we do not discriminate the weight of event element and event terms. In Experiment 7, we also did not discriminate the weight of event element and event terms. It is fair to compare Experiment 7 with Experiment 1 and it's unfair to compare Experiment 7 with Experiment 3. It looks like the relevance between nodes (event terms or event elements) can help to improve the performance. However, performance of both dependent and independent event-based summarization need to be improved further, compared with human performance in Experiment 8.

6 Discussion

As discussed in Section 2, event-based approaches are also employed in previous works. We evaluate our work in this context. As event-based approaches in this paper are similar with that of Filatovia and Hatzivassiloglou (2004), and the evaluation data set is the same one, the results are compared with theirs.

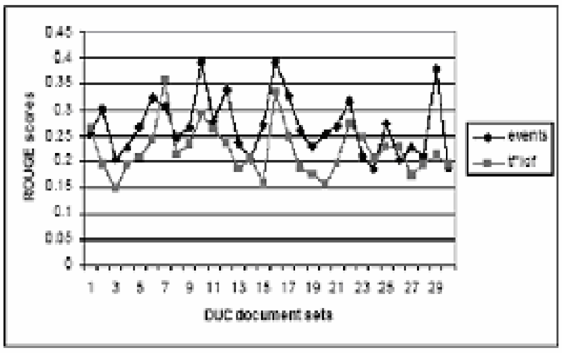


Figure 2. Results reported in (Filatovia and Hatzivassiloglou 2004)

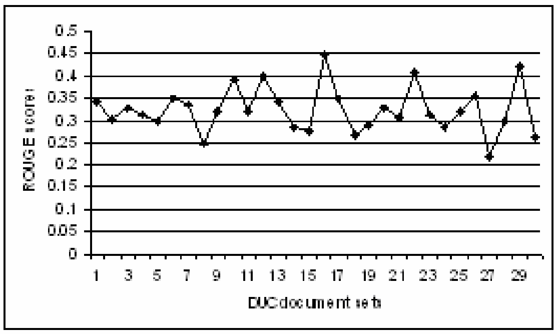


Figure 3. Results of relevant event-based approach

Filatovia and Hatzivassiloglou (2004) report the ROUGE scores according to each cluster of DUC 2001 data collection in Figure 2. In this figure, the bold line represents their event-based approach and the light line refers to tf*idf approach. It can be seen that the event-based approach performs better. The evaluation of the relevant event-based approach presented this paper is shown in Figure 3. The proposed approach achieves significant improvement on most document clusters. The reason seems that the relevance between events is exploited.

Centroid is a successful term-based summarization approach. For caparison, we employ MEAD (Radev et.al., 2004) to generate Centroid-based summaries. Results show that Centroid is better than our relevant event-based approach. After comparing the summaries given by the two approaches, we found some limitation of our approach.

Event-based approach does not work well on documents with rare events. We plan to discriminate the type of documents and apply event-based approach on suitable documents. Our relevant event-based approach is instance-based and too sensitive to number of instances of entities. Concepts seem better to represent meanings of events, as they are really things we care about. In the future, the event map will be build based on concepts and relationships between them. External knowledge may be exploited to refine this concept map.

7 Conclusion

In this study, we investigated generic summarization. An event-based scheme was employed to represent document and identify important content. The independent event-based approach identified important content according to event frequency. We also investigated the different importance of event terms in different context. Experiment showed that this idea achieved promising results. Then we explored summarization under different length limitation. We found that our independent event-based approaches acted well with longer summaries.

In the relevant event-based approach, events were linked together by same or similar event terms and event elements. Experiments showed that the relevance between events can improve the performance of summarization. Compared with close related work, we achieved encouraging improvement.

References

Regina Barzilay, and Michael Elhadad. 1997. Using lexical chains for text summarization. In Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization, 10-17.

Ronald Brandow, Karl Mitze, and Lisa F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. Information Processing and Management 31(5):675-686.

John M. Conroy and Judith D. Schlesinger. 2004. Left-brain/right-brain multi-document summarization. Available at http://duc.nist.gov/pubs.html

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan. 2002. GATE: a framework and graphical development environment for robust NLP tools and applications. In Proceedings of the 40th Annual Meeting of the Association for computational Linguistics (ACL'02).

Gerald Francis DeJong. 1978. Fast skimming of news stories: the FRUMP system. Ph.D. thesis, Yale University.

H.P. Edmundson. 1969. New methods in automatic extracting. Journal of the Association for computing machinery, 16(2):264-285.

Elena Filatova and Vasileios Hatzivassiloglou. Event-based extractive summarization. 2004. In Proceedings of the ACL-04 Workshop, 104-111.

Julian Kupiec, Jan Pedersen and Francine Chen. 1995. A trainable document summarizer. In Proceedings of the 18th ACM-SIGIR conference, 68-73.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In Proceedings of HLT-NAACL, Edmonton, Canada, May.

H.P. Luhn. 1958. The automatic creation of literature abstracts. IBM Journal of Research and Development 2:159-165.

Daniel Marcu. 1997. The rhetorical parsing of natural language texts. In Proceedings of the 35th Annual Meeting of the Association for computational Linguistics (ACL'97), 96-103.

Dragomir R. Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, Zhu Zhang. 2004. MEAD - a platform for multidocument multilingual text summarization. LREC 2004.

Simone Teufel and Marc Moens. 1999. Argumentative classification of extracted sentences as a first step towards flexible abstracting. Advances in Automatic Text Summarization, Inderjeet Mani and Mark T. Maybury (editors), 137-154. Cambridge, Massachusetts: MIT Press.

Larry Page, Sergey Brin, et al. 1998. The PageRank Citation Ranking: Bring Order to the Web. Technical Report, Stanford University, 1998.

Lucy Vanderwende, Michele Banko, and Arul Menezes. 2004. Event-centric summary generation. Available at http://duc.nist.gov/pubs.html

Masaharu Yoshioka and Makoto Haraguchi. 2004. Multiple news articles summarization based on event reference information. In Working Notes of the Fourth NTCIR Workshop Meeting, National Institute of Informatics, 2004.

Discursive Usage of Six Chinese Punctuation Marks

YUE Ming
Department of Applied Linguistics
Communication University of China
100024 Beijing, China
yueming@cuc.edu.cn

Abstract

Both rhetorical structure and punctuation have been helpful in discourse processing. Based on a corpus annotation project, this paper reports the discursive usage of 6 Chinese punctuation marks in news commentary texts: Colon, Dash, Ellipsis, Exclamation Mark, Question Mark, and Semicolon. The rhetorical patterns of these marks are compared against patterns around cue phrases in general. Results show that these Chinese punctuation marks, though fewer in number than cue phrases, are easy to identify, have strong correlation with certain relations, and can be used as distinctive indicators of nuclearity in Chinese texts.

1 Introduction

Rhetorical structure has been proven useful in NLP projects such as text generation, summarization, machine translation and essay scoring. Automatic discourse parsing remains an elusive task, however, despite much rule-based research on lexical cues such as anaphora and conjunctions. Parsing through machine learning has encountered a bottleneck, due to limited resources--there is only one English RST treebank publicly available, and one RST-annotated German corpus on its way.

Punctuation marks (PMs) have been proven useful in RST annotation as well as in many other NLP tasks such as Part-of-Speech tagging, Word Sense Disambiguation, Near-duplicate detection, bilingual alignment (e.g. Chuang and Yeh, 2005), etc. Dale (1991) noticed the role of PMs in determining rhetorical relations. Say (1998) did a study on their roles in English discourse structure. Marcu (1997) and Corston-Oliver (1998) based their automatic discourse parser partially on PMs and other orthographical cues. Tsou et al. (1999) and Chan et al. (2000) use PMs to disambiguate candidate Discourse Markers for a Chinese summarization system. Reitter (2003) also used PMs to distinguish ATTRIBUTION and ELABORATION relations in his Feature-rich SVM rhetorical analysis system.

All these inspired us to survey on the rhetorical patterns around Chinese PMs, so as to provide more direct a priori scores for the coarse rhetorical analyzer by Zhang et al. (2000) in their hybrid summarization system.

This paper is organized into 5 parts: Section 2 gives an overview of a Chinese RST treebank under construction, and a survey on the syntax of six main PMs in the corpus: Colon, Dash, Ellipses, Exclamation Mark, Question Mark, and Semicolon. Section 3 reports rhetorical patterns around these PMs. Section 4 is a discussion on the effectiveness of these PMs in comparison with Chinese cue phrases. Section 5 is a summary and Section 6 directions for future work.

2 Overview of Chinese RST treebank under construction

2.1 Corpus data

For the purpose of language engineering and linguistic investigation, we are constructing a Chinese corpus comparable to the English WSJ-RST treebank and the German Potsdam Commentary Corpus (Carlson et al. 2003; Stede 2004). Texts in our corpus were downloaded from the official website of *People's Daily*[1], where important *Caijingpinlun*[2] (CJPL) articles

[1] www.people.com.cn.
[2] *Caijinpinglun* (CJPL) in Chinese means "financial and business commentary", and usually covers various topics in social economic life, such as fiscal policies, financial reports,

by major media entities were republished. With over 400 authors and editors involved, our texts can be regarded as a good indicator of the general use of Chinese by Mainland native speakers.

At the moment our CJPL corpus has a total of 395 texts, 785,045 characters, and 84,182 punctuation marks (including pruned spaces). Although on average there are 9.3 characters between every two marks, sentences in CJPL are long, with 51.8 characters per common sentence delimiters (Full Stop, Question Mark and Exclamation Mark).

2.2 Segmentation

We are informed of the German Potsdam Commentary Corpus construction, in which they (Reitter 2003) designed a program for automatic segmentation at clausal level after each Sign="$."(including {., ?, !, ;, :, …}) and Sign="$,"(including {,})[3]. Human interference with the segmentation results was not allowed, but annotators could retie over-segmented bits by using the JOINT relation.

Given the workload of discourse annotation, we decided to design a similar segmentation program. So we first normalized different encoding systems and variants of PMs (e.g. Dashes and Ellipses of various lengths), and then conducted a survey on the distribution (Fig. 1) and syntax of major Chinese punctuation marks (e.g. syntax of Chinese Dash in Table 1).

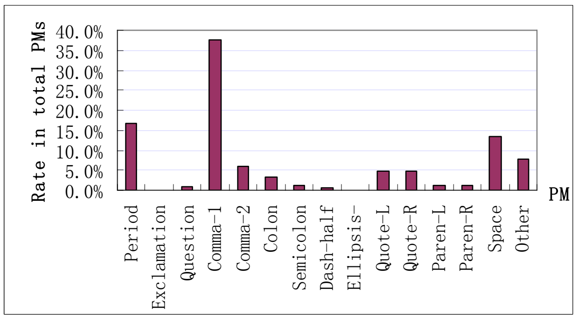


Figure 1: Percentage of major punctuation marks in the Chinese corpus[4]

C-Comma-1 is the most frequently used PM in the Chinese corpus. While it does delimit clauses, a study on 200 randomly selected C-Comma-1 tokens in our corpus shows that 55 of them are used after an independent NP or discourse marker. This rate, times the total number of C-Comma-1, means we would have to retie a huge number of over-segmented elements. So we decided not to take C-Comma-1 as a delimiter of our Elementary Unit of Discourse Analysis (EUDA) for the present.

Structure of C-——[5]	**%**
[NP+——NP+]NP	3.12%
[s+——s+]NP	0.44%
S*[NP——NP——VP]S	1.78%
S*[NP——s——VP]S	0.89%
S*[s——s——s]S	6.22%
<title>s+——Source：s+</title>	2.67%
<title>Source：s——s+</title>	0.44%
<para>S*s——</para>	1.33%
<para>S——S+</para>	2.22%
<para>S*s"——s+</para>	7.56%
<para>——S+</para>	12.44%
<para>S*s——s+</para>	60.89%
TTL	**100.00%**

Table 1: Syntax of Chinese Dash

42.9% of the colons in CJPL are used in the structural elements[6] of the texts. Other than these, 56.5% of the colons are used between clausal strings, only 0.6% of the colons are used after non-clausal strings.

99.6% instances of Exclamation Mark, Question Mark, Dash, Ellipses and Semicolon in the Chinese corpus are used after clausal strings.

In our corpus, 4.3% of the left quotation marks do not have a right match to indicate the end of a quote. Because many articles do not give clear indications of direct or indirect quotes[7], it is very difficult for the annotator to makeup.

Parentheses and brackets have a similar problem, with 3.2% marks missing their matches.

trading, management, economic conferences, transportation, entertainment, education, etc.

Collected by professional editors, most texts in our corpus are commentaries; some are of marginal genres by the Chinese standards.

[3] Dash, as a Sign= "$(", was not selected as a unit delimiter in the Potsdam Commentary Corpus.

[4] PMs are counted by individual symbols.

[5] The symbol "S" donates sentences with a common end mark, while "s" denotes structures orthographically end with one of the PMs studied here. "+" means one or more occurrences, "*" means zero or more occurrences. The category after a bracket pair indicates the syntactic role played by the unit enclosed, for example "[……]NP" means the ellipses functions as an NP within a clausal structure. "<para></para>" denotes paragraph opening and ending.

[6] By "Structural elements" we mean documentary information, such as Publishing Date, Source, Link, Editor, etc. Although these are parts of a news text, they are not the article proper, on which we annotate rhetorical relations.

[7] After a comparative study on the rhetorical structure of news published by some Hong Kong newspapers in both English and Chinese, Scollon and Scollon (1997) observed that "quotation is at best ambiguous in Chinese. No standard practice has been observed across newspapers in this set and even within a newspaper, it is not obvious which portions of the text are attributed to whom." We notice that Mainland newspapers have a similar phenomenon.

Besides, 53.9% of the marks appear in structural elements that we didn't intend to analyze[8].

Finally, we decided to use Period, the End-of-line symbol, and these six marks (Question Mark, Exclamation Mark, Colon, Semicolon, Ellipsis and Dash) as delimiters of our EUDA. Quotation mark, Parentheses, and Brackets were not selected.

A special program was designed to conduct the segmentation after each delimiter, with proper adjustment in cases when the delimiter is immediately followed by a right parenthesis, a right quotation mark, or another delimiter.

A pseudo-relation, SAME-UNIT, has been used during annotation to re-tie any discourse segment cut by the segmentation program into fragments.

2.3 Annotation and Validity Control

We use O'Donnell's RSTTool V3.43[9] as our annotation software. We started from the Extended-RST relation set embedded in the software, adding gradually some new relations, and finally got an inventory of 47 relations. We take the same rhetorical predicate with switched arguments as different relations, for instance, SOLUTIONHOOD-S, SOLUTIONHOOD-M and SOLUTIONHOOD-N are regarded as 3 relations.

Following Carlson et al. (2001) and Marcu's (1999) examples, we've composed a 60-page Chinese RST annotation manual, which includes preprocessing procedures, segmentation rules, definitions and examples of the relations, tag definitions for structural elements, tagging conventions for special structures, and a relation selection protocol. When annotating, we choose the most indicative relation according to the manual. Trees are constructed with binary branches except for multinuclear relations.

One experienced annotator had sketched trees for all the 395 files before the completion of the manual. Then she annotated 97 shortest files from 197 randomly selected texts, working independently and with constant reference to the manual. After a one-month break, she re-annotated the 97 files, with reference to the manual and with occasional consultation with Chinese journalists and linguists. The last version, though far from error-free, is currently taken as *the right* version for reliability tests and other statistics.

An intra-coder accuracy test has bee taken between the 1st and 2nd versions of 97 finished trees. The intra-coder accuracy rate (R_v) for a particular variable is defined as

$$R_v = \frac{2*(AT-AS)}{TT-TS} *100\%$$

Where

AT= number of agreed tags;

TT= number of total tags;

TS= number of total tags for structural elements;

AS= number of agreed tags for structural elements.

R_r for relation tags is 84.39%, R_u for unit tags is 85.61%, and R_n for nuclearity tags is 88.12%.

Because SPSS can only calculate Kappa Coefficient for symmetric data, we've only measured Kappa for relation tags to the EUDAs. The outcome, K_r=.738, is quite high.

3 Results

The 97 double-annotated files have in the main body of their texts a total of 677 paragraphs and 1,914 EUDAs. Relational patterns of those PMs are reported in Table 2-7 below[10]. The "N", "S" or "M" tags after each relation indicate the nuclearity status of each EUDA ended with a certain PM. The number of those PMs used in structural elements of CJPL texts are also reported as they make up the total percentage.

Relation (C-？)	P(r|pm)	P(pm|r)
Antithesis-N	1.14%	2.70%
Background-N	2.27%	3.39%
Concession-N	7.95%	7.29%
Conjunction-M	30.68%	5.24%
Disjunction-M	4.55%	36.36%
Elaboration-N	2.27%	1.10%
Elaboration-S	2.27%	1.10%
Evaluation-N	1.14%	0.72%
Interpretation-N	1.14%	0.67%
Joint-M	4.55%	6.90%
Justify-N	4.55%	1.75%
Justify-S	4.55%	1.75%
Nonvolitional-cause-S	2.27%	1.43%
Nonvolitional-result-S	1.14%	0.71%
Otherwise-S	1.14%	16.67%
Solutionhood-M	4.55%	5.33%
Solutionhood-S	14.78%	17.33%
Volitional-cause-N	1.14%	1.32%
Structural elements	7.96%	0.99%
TTL	**100.00%**	**N/A**

Table 2: Rhetorical pattern of C-Question

[8] Parentheses, and other PMs used in structural elements of CJPL texts, are of high relevance to discourse parsing, since they can be used in a preprocessor to filter out text fragments that do not need be annotated in terms of RST.

[9] Publicly downloadable at www.wagsoft.com.

[10] Based on data from the 2nd version of annotated texts.

Relation (C-！)	P(r|pm)	P(pm|r)
Addition-S	5.26%	14.29%
Conjunction-M	15.79%	0.58%
Elaboration-S	5.26%	0.55%
Evaluation-S	10.53%	1.44%
Evidence-S	10.53%	2.33%
Joint-M	5.26%	1.72%
Justify-N	5.26%	0.44%
Justify-S	5.26%	0.44%
Nonvolitional-cause-N	5.26%	0.71%
Solutionhood-N	5.26%	1.33%
Volitional-cause-S	5.26%	1.32%
Structural elements	21.05%	0.57%
TTL	**100.00%**	**N/A**

Table 3: Rhetorical pattern of C-Exclamation

Relation (C-：)	P(r|pm)	P(pm|r)
Attribution-S	10.93%	68.00%
Background-N	0.64%	3.39%
Background-S	0.32%	1.69%
Concession-N	0.32%	1.04%
Elaboration-N	18.97%	32.42%
Evaluation-N	0.64%	1.44%
Justify-S	0.32%	0.44%
Nonvolitional-cause-N	0.32%	0.71%
Preparation-S	4.18%	13.40%
Same-unit-S	0.32%	4.35%
Volitional-cause-N	0.32%	1.32%
Structural elements	62.70%[11]	27.70%
TTL	**100.00%**	**N/A**

Table 4: Rhetorical pattern of C-Colon

Relation (C-；)	P(r|pm)	P(pm|r)
Antithesis-S	1.00%	2.70%
Background-N	1.00%	1.69%
Background-S	1.00%	1.69%
Conjunction-M	59.00%	11.46%
Contrast-M	7.00%	7.69%
Disjunction-M	2.00%	18.18%
List-M	23.00%	24.73%
Purpose-N	1.00%	6.67%
Same-unit-M	2.00%	8.70%
Sequence-M	3.00%	6.12%
TTL	**100.00%**	**N/A**

Table 5: Rhetorical pattern of C-Semicolon

Relation (C-……)	P(r|pm)	P(pm|r)
Conjunction-M	12.50%	0.19%
Disjunction-M	12.50%	9.09%
Elaboration-S	25.00%	1.10%
Evidence-S	25.00%	2.33%
Evaluation-N	12.50%	0.72%
Volitional-result-S	12.50%	1.32%
TTL	**100.00%**	**N/A**

Table 6: Rhetorical pattern of C-Ellipses

Relation (C-——)	P(r|pm)	P(pm|r)
Elaboration-N	32.00%	4.40%
Elaboration-S	4.00%	0.55%
Evaluation-N	12.00%	2.16%
Evaluation-S	4.00%	0.72%
Nonvolitional-cause-S	4.00%	0.71%
Nonvolitional-result-S	4.00%	0.71%
Otherwise-S	4.00%	16.67%
Preparation-N	4.00%	1.03%
Purpose-N	4.00%	6.67%
Restatement-N	4.00%	14.29%
Same-unit-M	24.00%	26.09%
TTL	**100.00%**	**N/A**

Table 7: Rhetorical pattern of C-Dash

The above data suggest at least the following:

1) There is no one-to-one mapping between any of PM studied and a rhetorical relation. But some PMs have dominant rhetorical usages.
2) C-Question Mark is not most frequently related with SOLUTIONHOOD, but with CONJUNCTION. That is because a high percentage of questions in our corpus are rhetorical and used in groups to achieve certain argumentative force.
3) C-Colon is most frequently related with ATTRIBUTION and ELABORATION, apart from its usage in structural elements.
4) C-Semicolon is overwhelmingly associated with multinuclear relations, particularly with CONJUNCTION.
5) C-Dash usually indicates an ELABORATION relation. But since it is often used in pairs, it is often bound to both the Nucleus and Satellite units of a relation.
6) 82.3% tokens of the six Chinese PMs are uniquely related to EUDAs of certain nucleus status in a rhetorical relation, taking even C-Dash into account.
7) The following relations have more than 10% of their instances related to one of the six PMs studied here: ADDITION, ATTRIBUTION, CONJUNCTION, DISJUNCTION, ELABORATION, LIST, OTHERWISE, PREPARTION, RESTATEMENT and SOLUTIONHOOD.
8) Chinese PMs are used somewhat differently from their German equivalents, Exclamation Mark for instance (Fig.2):

[11] This is higher than the overall 42.93% rate for colons used in structural elements, for we've only finished 97 shortest ones from the 197 randomly selected files.

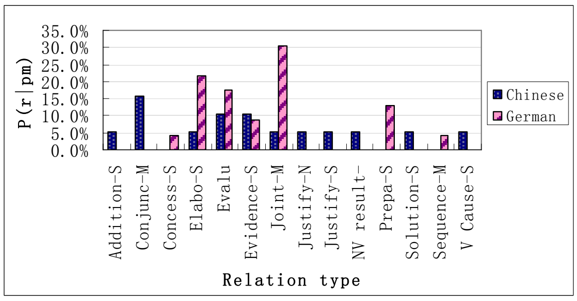


Figure 2: Rhetorical Function of Exclamation Mark in Chinese and German corpora

4 Discussion

How useful are these six PMs in the prediction of rhetorical relations in Chinese texts? In our opinion, this question can be answered partly through a comparison with Chinese cue phrases.

Cue phrases are widely discussed and exploited in the literature of both Chinese studies and RST applications as a major surface device. Unfortunately, Chinese cue phrases in natural texts are difficulty to identify automatically. As known, Chinese words are made up of 1, 2, or more characters, but there is no explicit word delimiter between any pair of adjacent words in a string of characters. Thus, they are not known before tokenization ("*fenci*" in Chinese, meaning "separating into words", or "word segmentation" so as to recognize meaningful words out of possible overlaps or combinations). The task may sound simple, but has been the focus of considerable research efforts (e.g. Webster and Kit, 1992; Guo 1997; Wu, 2003).

Since many cue phrases are made up of high-frequency characters (e.g. "而-*ER*" in "而-*er*" meaning "but/so/and", "然而-*ran'er*" meaning "but/however", "因而-*yin'er*" meaning "so/because of this", "而且-*erqie*" meaing "in addition" etc.; "此-*ci*" in "此后-*cihou*" meaning "later/hereafter", "因此-*yinci*" meaning "as a result", "由此看来-*youcikanlai*" meaning "on this ground/hence", etc.), a considerable amount of computation must be done before these cue phrases can ever been exploited.

Apart from tokenization, POS and WSD are other necessary steps that should be taken before making use of some common cue phrases. They are all hard nuts in Chinese language engineering. Interestingly, many researches done in these three areas have made use of the information carried by PMs (e.g. Sun et al. 1998).

Chan et al. (2000) did a study on identify Chinese connectives as signals of rhetorical relations for their Chinese summarizer. Their tests were successful. But like PMs, Chinese cue phrases are not in a one-to-one mapping relationship with rhetorical relations, either.

In our finished portion of CJPL corpus, we've identified 161 Types of cue phrases[12] at or above our EUDA level, recording 539 tokens. These cue phrases are scattered in 477 EDUAs, indicating 20.5% of the total relations in our finished portion of the corpus. Our six PMs, on the other hand, have 551 tokens in the same finished portion, delimiting 345 EUDAs (and 206 structural elements), and indicating 14.8% of the total relations. However, since there are far more types of cue phrases than types of punctuation marks, 90.1% of cue phrases are sparser at or above our EDUA level than the least frequently used PM—Ellipsis in this case.

And Chinese cue phrases don't signal all the rhetorical relations at all levels. For instance, CONJUNTION is the most frequently used relation in our annotated text (taking 22.1% of all the discursive relations), but it doesn't have strong correlation with any lexical item. Its most frequent lexical cue is "也-*ye*", taking 2.4%. ELABORATION is another common relation in CJPL, but it is rarely marked by cue phrases. ATTRIBUTION, SOLUTIONHOOD and DISJUNCTION are amongst other lowest marked relations in Chinese—they happen to be signaled quite significantly by a punctuation mark.

Given the cost to recognize Chinese cue phrases accurately, the sparseness of many of these cues, and the risk of missing all cue phrases for a particular discursive relation, punctuation marks with strong rhetorical preferences appear to be useful supplements to cue phrases.

5 Conclusion

Because rhetorical structure in Chinese texts is not explicit by itself, systematic and quantitative evaluation of various factors that can contribute to the automatic analysis of texts is quite necessary. The purpose of this study is to look into the discursive patterns of Chinese PMs, to see if they can facilitate discourse parsing without deep semantic analysis.

We have in this study observed the discursive usage of six Chinese PMs, from their overall distribution in our Chinese discourse corpus, their syntax in context, to their rhetorical roles at

[12] We are yet to give a theoretical definition of Cue Phrases in our study. But the identified ones range similarly to those English cue phrases listed in Marcu (1997).

or above our EUDA level. Current statistics seem to suggest clear patterns of their rhetorical roles, and their distinctive correlation with nuclearity in most relations. These patterns and correlation may be useful in NLP projects.

6 Future Work

We are conscious of the size and granularity of our treebank on which this analysis is based. We plan to get a larger team to work on the project, so as to make it more comparable to the English and German RST treebanks.

Since the distinctive nucleus status of EUDAs ended with these PMs may be useful in deciding growth point for RS-tree construction or for tree pruning in summarization, we are also interested in testing how well a baseline relation classifier performs if it always predicts the most frequent relations for these PMs.

Acknowledgement

Special thanks to Dr. Manfred Stede for licensing us to use the Potsdam Commentary Corpus. And thanks to Dr. Michael O'Donnell, FAN Taizhi, HU Fengguo, JIN Narisong, and MA Guangbin for their technical support. The author also fully appreciates the anonymous reviewers for their constructive comments.

References

Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual, *Technical Report ISI/TR-545*. www.isi.edu/~marcu.

Lynn Carlson, Daniel Marcu, and Mary. E. Okurowski. 2003. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In Jan van Kuppevelt and Ronnie Smith, editors, *Current Directions in Discourse and Dialogue*. Kluwer Academic Publishers. www.isi.edu/~marcu.

Samuel W. K. Chan, Tom B. Y. Lai, W. J. Gao and B. K. T'sou. 2000. Mining discourse markers for Chinese Textual Summarization. *Workshop on Automatic Summarization*, ACL 2000.

Thomas C. Chuang and Kevin C. Yeh. 2005. Aligning Parallel Bilingual Corpora Statistically with Punctuation Criteria. *Computational Linguistics and Chinese Language Processing*. Vol. 10, No. 1, March 2005, pp. 95-122.

Simon H. Corston-Oliver. 1998. Computing Representation of the Structure of Written Discourse. Technical Report. MSR-TR-98-15.

Robert Dale. 1991. The role of punctuation in discourse structure. *Working Notes for the AAAI Fall Symposium on Discourse Structure in Natural Language Understanding and Generation*. P13-13. Asilomar.

Jin GUO. 1997. Critical Tokenization and its Properties. *Computational Linguistics*, 23(4): 569-596.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Daniel Marcu. 1997. *The rhetorical parsing, summarization, and generation of natural language texts*. PhD thesis. University of Toronto. December 1997. www.isi.edu/~marcu

Daniel Marcu. 1999. *Instructions for manually annotating the discourse structures of texts.* www.isi.edu/~marcu

David Reitter. 2003. *Rhetorical Analysis with Rich-Feature Support Vector Models*. University of Potsdam, Diploma thesis in computational linguistics.

Bilge Say. 1998. *An Information-Based Approach to Punctuation*. Ph.D. dissertation, Bilkent University, Ankara, Turkey. http://www.cs.bilkent.edu.tr/~say/bilge.html.

Ron Scollon and Suzanne Wong Scollon. 1997. Point of view and citation: Fourteen Chinese and English versions of the 'same' news story. *Text*, 17 (1), 83-125.

Manfred Stede. 2004. The Potsdam Commentary Corpus. In *Proceedings of the ACL 2004 Workshop 'Discourse Annotation'*. Barcelona.

SUN Maosong, Dayang SHEN, and Benjamin K. Tsou, 1998. Chinese word segmentation without using lexicon and hand-crafted training data. In *Proceedings of COLING-ACL'98*.

Benjamin K.Tsou, Weijun Gao, T.V.Y Lai and S.W.K. Chan. 1999. Applying machine learning to identify Chinese discourse markers. *Proceedings of 1999 International Conference on Information Intelligence and Systems*. p 548-53, 31 Oct.-3 Nov. 1999 , Bethesda, MD, USA.

Jonathan J. Webster and Chunyu Kit. 1992. Tokenization as the initial phase in NLP. In *Proceedings of the 14th International Conference on Computational Linguistics* (COLING'92), pages 1,106-1,110, Nantes, France.

WU Andi. 2003. Chinese Word Segmentation in MSR-NLP. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, Sapporo, Japan.

ZHANG Yimin, LU Ru-Zhan and SHEN Li-Bin. 2000. A hybrid method for automatic Chinese discourse structure analysis. *Journal of Software*, v 11, n 11, Nov. 2000, p 1527-33.

Integrated Morphological and Syntactic Disambiguation for Modern Hebrew

Reut Tsarfaty
Institute for Logic, Language and Computation, University of Amsterdam
Plantage Muidergratch 24, 1018 TV Amsterdam, The Netherlands
rtsarfat@science.uva.nl

Abstract

Current parsing models are not immediately applicable for languages that exhibit strong interaction between morphology and syntax, e.g., Modern Hebrew (MH), Arabic and other Semitic languages. This work represents a first attempt at modeling morphological-syntactic interaction in a generative probabilistic framework to allow for MH parsing. We show that morphological information selected in tandem with syntactic categories is instrumental for parsing Semitic languages. We further show that redundant morphological information helps syntactic disambiguation.

1 Introduction

Natural Language Processing is typically viewed as consisting of different layers,[1] each of which is handled separately. The structure of Semitic languages poses clear challenges to this traditional division of labor. Specifically, Semitic languages demonstrate strong interaction between morphological and syntactic processing, which limits the applicability of standard tools for, e.g., parsing.

This work focuses on MH and explores the ways morphological and syntactic processing interact. Using a morphological analyzer, a part-of-speech tagger, and a PCFG-based general-purpose parser, we segment and parse MH sentences based on a small, annotated corpus. Our integrated model shows that percolating morphological ambiguity to the lowest level of non-terminals in the syntactic parse tree improves parsing accuracy. Moreover, we show that morphological cues facilitate syntactic disambiguation. A particular contribution of this work is to demonstrate that MH statistical parsing is *feasible*. Yet, the results obtained are not comparable to those of, e.g., state-of-the-art models for English, due to remaining syntactic ambiguity and limited morphological treatment. We conjecture that adequate morphological and syntactic processing of MH should be done in a unified framework, in which both levels can interact and share information in both directions.

Section 2 presents linguistic data that demonstrate the strong interaction between morphology and syntax in MH, thus motivating our choice to treat both in the same framework. Section 3 surveys previous work and demonstrates again the unavoidable interaction between the two. Section 4.1 puts forward the formal setting of an integrated probabilistic language model, followed by the evaluation metrics defined for the integrated task in section 4.2. Sections 4.3 and 4.4 then describe the experimental setup and preliminary results for our baseline implementation, and section 5 discusses more sophisticated models we intend to investigate.

2 Linguistic Data

Phrases and sentences in MH, as well as Arabic and other Semitic languages, have a relatively free word order.[2] In figure 1, for example, two distinct syntactic structures express the same grammatical relations. It is typically morphological information rather than word order that provides cues for structural dependencies (e.g., agreement on gender and number in figure 1 reveals the subject-predicate dependency).

[1]E.g., phonological, morphological, syntactic, semantic and pragmatic.

[2]MH allows for both SV and VS, and in some circumstances also VSO, SOV and others.

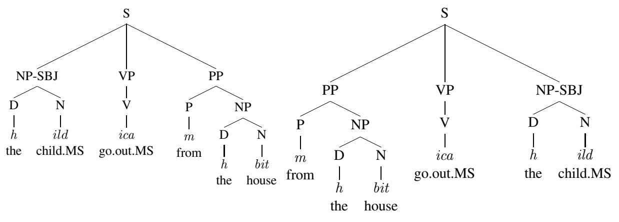


Figure 1: Word Order in MH Phrases (marking the agreement features M(asculine), S(ingular))

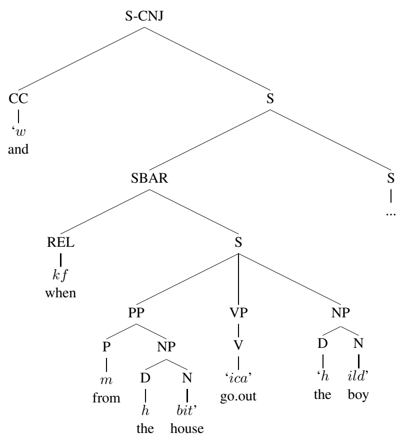


Figure 2: Syntactic Structures of MH Phrases (marking word boundaries with ' ')

Furthermore, boundaries of constituents in the syntactic structure of MH sentences need not coincide with word boundaries, as illustrated in figure 2. A MH word may coincide with a single constituent, as in '*ica*'[3] (go out), it may overlap with an entire phrase, as in '*h ild*' (the boy), or it may span across phrases as in '*w kf m h bit*' (and when from the house). Therefore, we conclude that in order to perform syntactic analysis (parsing) of MH sentences, we must first identify the morphological constituents that form MH words.

There are (at least) three distinct morphological processes in Semitic languages that play a role in word formation. *Derivational morphology* is a non-concatenative process in which verbs, nouns, and adjectives are derived from (tri-)consonantal roots plugged into templates of consonant/vowel skeletons. The word-forms in table 1, for example, are all derived from the same root, $[i][l][d]$ (child, birth), plugged into different templates. In addition, MH has a rich array of agreement features, such as gender, number and person, expressed in the word's *inflectional morphology*. Verbs, adjectives, determiners and numerals must agree on the inflectional features with the noun they comple-

[3]We adopt the transliteration of (Sima'an et al., 2001).

a.	b.	c.
'*ild*'	'*iild*'	'*mwld*'
$[i]e[l]e[d]$	$[i]i[l](l)e[d]$	$mw[][l](l)a[d]$
child	deliver a child	innate

Table 1: Derivational Morphology in MH ([..] mark templates' slots for consonantal roots, (..) mark obligatory doubling of roots' consonants.)

a.	b.
ild gdwl	*ildh gdwlh*
child.MS big.MS	child.FS big.FS
a big boy	a big girl

Table 2: Inflectional Morphology in MH (marking M(asculine)/F(eminine), S(ingular)/P(lural))

ment or modify. It can be seen in table 2 that the suffix h alters the noun '*ild*' (child) as well as its modifier '*gdwl*' (big) to feminine gender. Finally, particles that are prefixed to the word may serve different syntactic functions, yet a multiplicity of them may be *concatenated* together with the stem to form a single word. The word '*wkfmhbit*' in figure 2, for instance, is formed from a conjunction w (and), a relativizer *kf* (when), a preposition m (from), a definite article h (the) and a noun *bit* (house). Identifying such particles is crucial for analyzing syntactic structures as they reveal structural dependencies such as subordinate clauses, adjuncts, and prepositional phrase attachments.

At the same time, MH exhibits a large-scale ambiguity already at the word level, which means that there are multiple ways in which a word can be broken down to its constituent morphemes. This is further complicated by the fact that most vocalization marks (diacritics) are omitted in MH texts. To illustrate, table 3 lists two segmentation possibilities, four readings, and five meanings of different morphological analyses for the word-form '*fmnh*'.[4] Yet, the morphological analysis of a word-form, and in particular its morphological segmentation, cannot be disambiguated without reference to context, and various morphological features of syntactically related forms provide useful hints for morphological disambiguation. Figure 3 shows the correct analyses of the form '*fmnh*' in different syntactic contexts. Note that the correct analyses maintain agreement on gender and number between the noun and its modifier. In particular, the analysis 'that counted' (b)

[4]A statistical study on a MH corpus has shown that the average number of possible analyses per word-form was 2.1, while 55% of the word-forms were morphologically ambiguous (Sima'an et al., 2001).

'*fmnh*'	'*fmnh*'	'*fmnh*'	'*fmnh*'	'*f* + *mnh*'
shmena	shamna	shimna	shimna	she + mana
fat.FS	got-fat.FS	put-oil.FS	oil-of.FS	that + counted
fat (adj)	got fat (v)	put-oil (v)	her oil (n)	that (rel) counted (v)

Table 3: Morphological Analyses of the Word-form '*fmnh*'

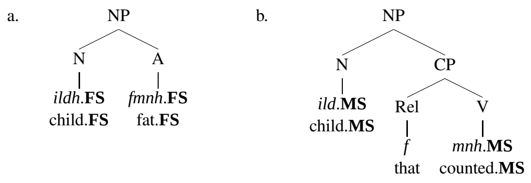


Figure 3: Ambiguity Resolution in Different Syntactic Contexts

is easily disambiguated, as it is the only one maintaining agreement with the modified noun.

In light of the above, we would want to conclude that syntactic processing must precede morphological analysis; however, this would contradict our previous conclusion. For this reason, independent morphological and syntactic analyzers for MH will not suffice. We suggest performing morphological and syntactic processing of MH utterances in a single, integrated, framework, thereby allowing shared information to support disambiguation in multiple tasks.

3 Related Work

As of yet there is no statistical parser for MH. Parsing models have been developed for different languages and state-of-the-art results have been reported for, e.g., English (Collins, 1997; Charniak, 2000). However, these models show impoverished morphological treatment, and they have not yet been successfully applied for MH parsing. (Sima'an et al., 2001) present an attempt to parse MH sentences based on a small, annotated corpus by applying a general-purpose Tree-gram model. However, their work presupposes correct morphological disambiguation prior to parsing.[5]

In order to treat morphological phenomena a few stand-alone morphological analyzers have been developed for MH.[6] Most analyzers consider words in isolation, and thus propose multiple analyses for each word. Analyzers which also attempt disambiguation require contextual information from surrounding word-forms or a shallow parser (e.g., (Adler and Gabai, 2005)).

A related research agenda is the development of part-of-speech taggers for MH and other Semitic languages. Such taggers need to address the segmentation of words into morphemes to which distinct morphosyntactic categories can be assigned (cf. figure 2). It was illustrated for both MH (Bar-Haim, 2005) and Arabic (Habash and Rambow, 2005) that an integrated approach towards making morphological (segmentation) and syntactic (POS tagging) decisions within the same architecture yields excellent results. The present work follows up on insights gathered from such studies, suggesting that an integrated framework is an adequate solution for the apparent circularity in morphological and syntactic processing of MH.

4 The Integrated Model

As a first attempt to model the interaction between the morphological and the syntactic tasks, we incorporate an intermediate level of *part-of-speech (POS) tagging* into our model. The key idea is that POS tags that are assigned to morphological segments at the word level coincide with the lowest level of non-terminals in the syntactic parse trees (cf. (Charniak et al., 1996)). Thus, POS tags can be used to pass information between the different tasks yet ensuring agreement between the two.

4.1 Formal Setting

Let w_1^m be a sequence of words from a fixed vocabulary, s_1^n be a sequence of segments of words from a (different) vocabulary, t_1^n a sequence of morphosyntactic categories from a finite tag-set, and let π be a syntactic parse tree.

We define *segmentation* as the task of identifying the sequence of morphological constituents that were concatenated to form a sequence of words. Formally, we define the task as (1), where $seg(w_1^m)$ is the set of segmentations resulting from all possible morphological analyses of w_1^n.

$$s_1^{n*} = \underset{s_1^n \in seg(w_1^m)}{\text{argmax}} \; P(s_1^n | w_1^m) \qquad (1)$$

Syntactic analysis, *parsing*, identifies the structure of phrases and sentences. In MH, such tree structures combine segments of words that serve different syntactic functions. We define it formally as (2), where $yield(\pi')$ is the ordered set of leaves of a syntactic parse tree π'.

$$\pi^* = \underset{\pi \in \{\pi' : yield(\pi') = s_1^n\}}{\text{argmax}} \; P(\pi | s_1^n) \qquad (2)$$

[5]The same holds for current work on parsing Arabic.
[6]Available at `mila.cs.technion.ac.il`.

Similarly, we define *POS tagging* as (3), where $analysis(s_1^n)$ is the set of all possible POS tag assignments for s_1^n.

$$t_1^{n*} = \underset{t_1^n \in analyses(s_1^n)}{\text{argmax}} \; P(t_1^n|s_1^n) \qquad (3)$$

The task of the *integrated model* is to find the most probable segmentation *and* syntactic parse tree given a sentence in MH, as in (4).

$$\langle \pi, s_1^n \rangle^* = \underset{\langle \pi, s_1^n \rangle}{\text{argmax}} P(\pi, s_1^n | w_1^m) \qquad (4)$$

We reinterpret (4) to distinguish the morphological and syntactic tasks, conditioning the latter on the former, yet maximizing for both.

$$\langle \pi, s_1^n \rangle^* = \underset{\langle \pi, s_1^n \rangle}{\text{argmax}} \underbrace{P(\pi|s_1^n, w_1^m)}_{\textit{parsing}} \underbrace{P(s_1^n|w_1^m)}_{\textit{segmentation}} \qquad (5)$$

Agreement between the tasks is implemented by incorporating morphosyntactic categories (POS tags) that are assigned to morphological segments and constrain the possible trees, resulting in (7).

$$\langle \pi, t_1^n, s_1^n \rangle^* = \underset{\langle \pi, t_1^n, s_1^n \rangle}{\text{argmax}} P(\pi, t_1^n, s_1^n | w_1^m) \qquad (6)$$

$$= \underset{\langle \pi, t_1^n, s_1^n \rangle}{\text{argmax}} \underbrace{P(\pi|t_1^n, s_1^n, w_1^m)}_{\textit{parsing}} \underbrace{P(t_1^n|s_1^n, w_1^m)}_{\textit{tagging}} \underbrace{P(s_1^n|w_1^m)}_{\textit{segmentation}} \qquad (7)$$

Finally, we employ the assumption that $P(w_1^m|s_1^n) \approx 1$, since segments can only be conjoined in a certain order.[7] So, instead of (5) and (7) we end up with (8) and (9), respectively.

$$\approx \underset{\langle \pi, s_1^n \rangle}{\text{argmax}} \underbrace{P(\pi|s_1^n)}_{\textit{parsing}} \underbrace{P(s_1^n|w_1^m)}_{\textit{segmentation}} \qquad (8)$$

$$\approx \underset{\langle \pi, t_1^n, s_1^n \rangle}{\text{argmax}} \underbrace{P(\pi|t_1^n, s_1^n)}_{\textit{parsing}} \underbrace{P(t_1^n|s_1^n)}_{\textit{tagging}} \underbrace{P(s_1^n|w_1^m)}_{\textit{segmentation}} \qquad (9)$$

4.2 Evaluation Metrics

The intertwined nature of morphology and syntax in MH poses additional challenges to standard parsing *evaluation metrics*. First, note that we cannot use morphemes as the basic units for comparison, as the proposed segmentation need not coincide with the gold segmentation for a given sentence. Since words are complex entities that can span across phrases (see figure 2), we cannot use them for comparison either. We propose to redefine *precision* and *recall* by considering the spans of syntactic categories based on the (space-free) sequences of characters to which they correspond. Formally, we define syntactic constituents as $\langle i, A, j \rangle$ where i, j mark the location of characters. $T = \{\langle i, A, j \rangle | A \textit{ spans from } i \textit{ to } j\}$ and $G = \{\langle i, A, j \rangle | A \textit{ spans from } i \textit{ to } j\}$ represent the test/gold parses, respectively, and we calculate:[8]

$$\textit{Labeled Precision} = \#(G \cap T)/\#T \qquad (10)$$

$$\textit{Labeled Recall} = \#(G \cap T)/\#G \qquad (11)$$

4.3 Experimental Setup

Our departure point for the syntactic analysis of MH is that the basic units for processing are not words, but morphological segments that are concatenated together to form words. Therefore, we obtain a segment-based probabilistic grammar by training a Probabilistic Context Free Grammar (PCFG) on a segmented and annotated MH corpus (Sima'an et al., 2001). Then, we use existing tools — i.e., a morphological analyzer (Segal, 2000), a part-of-speech tagger (Bar-Haim, 2005), and a general-purpose parser (Schmid, 2000) — to find compatible morphological segmentations and syntactic analyses for unseen sentences.

The Data The data set we use is taken from the MH treebank which consists of 5001 sentences from the daily newspaper 'ha'aretz' (Sima'an et al., 2001). We employ the syntactic categories and POS tag sets developed therein. Our data set includes 3257 sentences of length greater than 1 and less than 21. The number of segments per sentence is 60% higher than the number of words per sentence.[9] We conducted 8 experiments in which the data is split to training and test sets and apply cross-fold validation to obtain robust averages.

The Models *Model I* uses the morphological analyzer and the POS tagger to find the most probable segmentation for a given sentence. This is done by providing the POS tagger with multiple morphological analyses per word and maximizing the sum $\sum_{t_1^n} P(t_1^n, s_1^n|w_1^m)$ (Bar-Haim, 2005, section 8.2). Then, the parser is used to find the most

[7]Since concatenated particles (conjunctions et al.) appear in front of the stem, pronominal and inflectional affixes at the end of the stem, and derivational morphology inside the stem, there is typically a unique way to restore word boundaries.

[8]Covert definite article errors are counted only at the POS tags level and discounted at the phrase-level.

[9]The average number of words per sentence in the complete corpus is 17 while the average number of morphological segments per sentence is 26.

probable parse tree for the selected sequence of morphological segments. Formally, this model is a first approximation of equation (8) using a step-wise maximization instead of a joint one.[10]

In *Model II* we percolate the morphological ambiguity further, to the lowest level of non-terminals in the syntactic trees. Here we use the morphological analyzer and the POS tagger to find the most probable segmentation *and* POS tag assignment by maximizing the joint probability $P(t_1^n, s_1^n|w_1^m)$ (Bar-Haim, 2005, section 5.2). Then, the parser is used to parse the tagged segments. Formally, this model attempts to approximate equation (9). (Note that here we couple a morphological and a syntactic decision, as we are looking to maximize $P(t_1^n, s_1^n|w_1^m) \approx P(t_1^n|s_1^n)P(s_1^n|w_1^m)$ and constrain the space of trees to those that agree with the resulting analysis.)[11]

In both models, *smoothing* the estimated probabilities is delegated to the relevant subcomponents. Out of vocabulary (OOV) words are treated by the morphological analyzer, which proposes all possible segmentations assuming that the stem is a proper noun. The Tri-gram model used for POS tagging is smoothed using Good-Turing discounting (see (Bar-Haim, 2005, section 6.1)), and the parser uses absolute discounting with various backoff strategies (Schmid, 2000, section 4.4).

The Tag-Sets To examine the usefulness of various morphological features shared with the parsing task, we alter the set of morphosyntactic categories to include more fine-grained morphological distinctions. We use three sets: *Set A* contains bare POS categories, *Set B* identifies also definite nouns marked for possession, and *Set C* adds the distinction between finite and non-finite verb forms.

Evaluation We use seven measures to evaluate our models' performance on the integrated task.

[10]At the cost of incurring indepence assumptions, a step-wise architecture is computationally cheaper than a joint one and this is perhaps the simplest end-to-end architecture for MH parsing imaginable. In the absence of previous MH parsing results, this model is suitable to serve as a baseline against which we compare more sophisticated models.

[11]We further developed a third model, *Model III*, which is a more faithful approximation, yet computationally affordable, of equation (9). There we percolate the ambiguity all the way through the integrated architecture by means of providing the parser with the n-best sequences of tagged morphological segments and selecting the analysis $\langle \pi, t_1^n, s_1^n \rangle$ which maximizes the production $P(\pi|t_1^n, s_1^n)P(s_1^n, t_1^n|w_1^m)$. However, we have not yet obtained robust results for this model prior to the submission of this paper, and therefore we leave it for future discussion.

	String Cover.	Labeled Prec. / Rec.	POS tags Prec. / Rec.	Segment. Prec. / Rec.
Model *I-A*	99.2%	60.3% / 58.4%	82.4% / 82.6%	94.4% / 94.7 %
Model *II-A*	95.9%	60.7% / 60.5%	84.5% / 84.8%	91.3% / 91.6%
Model *I-B*	99.2 %	60.3% / 58.4%	81.6% / 82.3%	94.2% / 95.0%
Model *II-B*	95.7%	60.7% / 60.5%	82.8% / 83.5%	90.9% / 91.7%
Model *I-C*	99.2%	60.9% / 59.2%	80.4% / 81.1%	94.2% / 95.1%
Model *II-C*	95.9%	61.7% / 61.9%	81.6% / 82.3%	91.0% / 91.9%

Table 4: Evaluation Metrics, Models *I* and *II*

First, we present the percentage of sentences for which the model could propose a pair of corresponding morphological and syntactic analyses. This measure is referred to as *string coverage*. To indicate morphological disambiguation capabilities we report *segmentation precision and recall*. To capture tagging and parsing accuracy, we refer to our redefined Parseval measures and separate the evaluation of morphosyntactic categories, i.e., *POS tags precision and recall*, and phrase-level syntactic categories, i.e., *labeled precision and recall* (where root nodes are discarded and empty trees are counted as zero).[12] The labeled categories are evaluated against the original tag set.

4.4 Results

Table 4 shows the evaluation scores for models *I-A* to *II-C*. To the best of our knowledge, these are the first parsing results for MH assuming no manual interference for morphological disambiguation.

For all sets, parsing of tagged-segments (*Model II*) shows improvement of up to 2% over parsing bare segments' sequences (*Model I*). This indicates that morphosyntactic information selected in tandem with morphological segmentation is more informative for syntactic analysis than segmentation alone. We also observe decreasing string coverage for *Model II*, possibly since disambiguation based on short context may result in a probable, yet incorrect, POS tag assignment for which the parser cannot recover a syntactic analysis. Correct disambiguation may depend on long-distance cues, e.g., agreement, so we advocate percolating the ambiguity further up to the parser.

Comparing the performance for the different tag sets, parsing accuracy increases for models *I-B/C* and *II-B/C* while POS tagging results decrease. These results seem to contradict the common wisdom that performance on a 'complex' task de-

[12]Since we evaluate the models' performance on an *integrated* task, sentences in which one of the subcomponents failed to propose an analysis counts as zero for *all* subtasks.

pends on a 'simpler', preceding one; yet, they support our thesis that morphological information orthogonal to syntactic categories facilitates syntactic analysis and improves disambiguation capacity.

5 Discussion

Devising a baseline model for morphological and syntactic processing is of great importance for the development of a broad-coverage statistical parser for MH. Here we provide a set of standardized baseline results for later comparison while consolidating the formal and architectural underpinning of an integrated model. However, our results were obtained using a relatively small set of training data and a weak (unlexicalized) parser, due to the size of the corpus and its annotated scheme.[13] Training a PCFG on our treebank resulted in a severely ambiguous grammar, mainly due to high phrase structure variability.

To compensate for the flat, ambiguous phrase-structures, in the future we intend to employ probabilistic grammars in which all levels of non-terminals are augmented with morphological information percolated up the tree. Furthermore, the MH treebank annotation scheme features a set of so-called *functional features*[14] which express grammatical relations. We propose to learn the correlation between various morphological markings and functional features, thereby constraining the space of syntactic structures to those which express meaningful predicate-argument structures.

Since our data set is relatively small,[15] introducing orthogonal morphological information to syntactic categories may result in severe data sparseness. In the current architecture, smoothing is handled separately by each of the subcomponents. Enriched grammars would allow us to exploit multiple levels of information in smoothing the estimated probabilities and to redistribute probability mass to unattested events based on their *similarity* to attested events in their integrated representation.

6 Conclusion

Traditional approaches for devising parsing models, smoothing techniques and evaluation metrics are not well suited for MH, as they presuppose separate levels of processing. Different languages mark regularities in their surface structures in different ways – English encodes regularities in word order, while MH provides useful hints about grammatical relations in its derivational and inflectional morphology. In the future we intend to develop more sophisticated models implementing closer interaction between morphology and syntax, by means of which we hope to boost parsing accuracy and improve morphological disambiguation.

Acknowledgments I would like to thank Khalil Sima'an for supervising this work, Remko Scha, Rens Bod and Jelle Zuidema for helpful comments, and Alon Itai, Yoad Winter and Shuly Wintner for discussion. The Knowledge Center for Hebrew Processing provided corpora and tools, and Roy Bar-Haim provided knowledge and technical support for which I am grateful. This work is funded by the Netherlands Organization for Scientific Research (NWO) grant 017.001.271.

References

Meni Adler and Dudi Gabai. 2005. Morphological Analyzer and Disambiguator for Modern Hebrew. Knowledge Center for Processing Hebrew.

Roy Bar-Haim. 2005. Part-of-Speech Tagging for Hebrew and Other Semitic Languages. Master's thesis, Technion, Haifa, Israel.

Eugene Charniak, Glenn Carroll, John Adcock, Anthony R. Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael L. Littman, and John McCann. 1996. Taggers for Parsers. *AI*, 85(1-2):45–57.

Eugene Charniak. 2000. A Maximum-Entropy-Inspired Parser. In *Proceedings of NAACL 2000*.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of ACL-EACL 1997*.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of ACL 2005*.

Helmut Schmid, 2000. *LoPar: Design and Implementation*. Institute for Computational Linguistics, University of Stuttgart.

Erel Segal. 2000. A Probabilistic Morphological Analyzer for Hebrew Undotted Texts. Master's thesis, Computer Science Department, Technion, Isreal.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a Tree-Bank for Modern Hebrew Text. In *Traitement Automatique des Langues*, volume 42, pages 347–380.

[13]The lack of head marking, for instance, precludes the use of lexicalized models à la (Collins, 1997).

[14]SBJ for subject, OBJ for object, COM for complement, etc. (Sima'an et al., 2001).

[15]The size of our treebank is less than 30% of the Arabic Treebank, and less than 10% of the WSJ Penn Treebank.

A Hybrid Relational Approach for WSD – First Results

Lucia Specia
Núcleo Interinstitucional de Lingüística Computational – ICMC – University of São Paulo
Caixa Postal 668, 13560-970, São Carlos, SP, Brazil
lspecia@icmc.usp.br

Abstract

We present a novel hybrid approach for Word Sense Disambiguation (WSD) which makes use of a relational formalism to represent instances and background knowledge. It is built using Inductive Logic Programming techniques to combine evidence coming from both sources during the learning process, producing a rule-based WSD model. We experimented with this approach to disambiguate 7 highly ambiguous verbs in English-Portuguese translation. Results showed that the approach is promising, achieving an average accuracy of 75%, which outperforms the other machine learning techniques investigated (66%).

1 Introduction

Word Sense Disambiguation (WSD) is concerned with the identification of the correct sense of an ambiguous word given its context. Although it can be thought of as an independent task, its importance is more easily realized when it is applied to particular tasks, such as Information Retrieval or Machine Translation (MT). In MT, the application we are focusing on, a WSD (or *translation disambiguation*) module should identify the correct translation for a source word when options with different meanings are available.

As shown by Vickrey et al. (2005), we believe that a WSD module can significantly improve the performance of MT systems, provided that such module is developed following specific requirements of MT, e.g., employing multilingual sense repositories. Differences between monolingual and multilingual WSD are very significant for MT, since it is concerned only with the ambiguities that appear in the translation (Hutchins and Sommers, 1992).

In this paper we present a novel approach for WSD, designed focusing on MT. It follows a hybrid strategy, i.e., knowledge and corpus-based, and employs a highly expressive relational formalism to represent both the examples and background knowledge. This approach allows the exploitation of several knowledge sources, together with evidences provided by examples of disambiguation, both automatically extracted from lexical resources and sense tagged corpora. This is achieved using Inductive Logic Programming (Muggleton, 1991), which has not been exploited for WSD so far. In this paper we investigate the disambiguation of 7 highly ambiguous verbs in English-Portuguese MT, using knowledge from 7 syntactic, semantic and pragmatic sources.

In what follows, we first present some related approaches on WSD for MT, focusing oh their limitations (Section 2). We then give some basic concepts on Inductive Logic Programming and describe our approach (Section 3). Finally, we present our initial experiments and the results achieved (Section 4).

2 Related work

Many approaches have been proposed for WSD, but only a few are designed for specific applications, such as MT. Existing multilingual approaches can be classified as (a) knowledge-based approaches, which make use of linguistic knowledge manually codified or extracted from lexical resources (Pedersen, 1997; Dorr and Katsova, 1998); (b) corpus-based approaches, which make use of knowledge automatically acquired from text using machine learning algorithms (Lee, 2002; Vickrey et al., 2005); and (c) hybrid approaches, which employ techniques from the two other approaches (Zinovjeva, 2000).

Hybrid approaches potentially explore the advantages of both other strategies, yielding accurate and comprehensive systems. However, they are quite rare, even in monolingual contexts (Stevenson and Wilks, 2001, e.g.), and they are not able to integrate and use knowledge coming from corpus and other resources during the learning process.

In fact, current hybrid approaches usually employ knowledge sources in pre-processing steps, and then use machine learning algorithms to combine disambiguation evidence from those sources. This strategy is necessary due to the limitations of the formalism used to represent examples in the machine learning process: the propositional formalism, which structures data in attribute-value vectors.

Even though it is known that great part of the knowledge regarding to languages is relational (e.g., syntactic or semantic relations among words in a sentence) (Mooney, 1997), the propositional formalism traditionally employed makes unfeasible the representation of substantial relational knowledge and the use of this knowledge during the learning process.

According to the attribute-value representation, one attribute has to be created for every feature, and the same structure has to be used to characterize all the examples. In order to represent the syntactic relations between every pair of words in a sentence, e.g., it will be necessary to create at least one attribute for each possible relation (Figure 1). This would result in an enormous number of attributes, since the possibilities can be many in distinct sentences. Also, there could be more than one pair with the same relation.

Sentence: *John gave to Mary a big cake.*			
verb$_1$-subj$_1$	verb$_1$-obj$_1$	mod$_1$-obj$_1$	…
give-john	give-cake	big-cake	…

Figure 1. Attribute-value vector for syntactic relations

Given that some types of information are not available for certain instances, many attributes will have null values. Consequently, the representation of the sample data set tends to become highly sparse. It is well-known that sparseness on data ensue serious problems to the machine learning process in general (Brown and Kros, 2003). Certainly, data will become sparser as more knowledge about the examples is considered, and the problem will be even more critical if relational knowledge is used.

Therefore, at least three relevant problems arise from the use of a propositional representation in corpus-based and hybrid approaches: (a) the limitation on its expressiveness power, making it difficult to represent relational and other more complex knowledge; (b) the sparseness in data; and (c) the lack of integration of the evidences provided by examples and linguistic knowledge.

3 A hybrid relational approach for WSD

We propose a novel hybrid approach for WSD based on a relational representation of both examples and linguistic knowledge. This representation is considerably more expressive, avoids sparseness in data, and allows the use of these two types of evidence during the learning process.

3.1 Sample data

We address the disambiguation of 7 verbs selected according to the results of a corpus study (Specia, 2005). To build our sample corpus, we collected 200 English sentences containing each of the verbs from a corpus comprising fiction books. In a previous step, each sentence was automatically tagged with the translation of the verb, part-of-speech and lemmas of all words, and subject-object syntactic relations with respect to the verb (Specia et al., 2005). The set of verbs, their possible translations, and the accuracy of the most frequent translation are shown in Table 1.

Verb	# Translations	Most frequent translation - %
come	11	50.3
get	17	21
give	5	88.8
go	11	68.5
look	7	50.3
make	11	70
take	13	28.5

Table 1. Verbs and their possible senses in our corpus

3.2 Inductive Logic Programming

We utilize Inductive Logic Programming (ILP) (Muggleton, 1991) to explore relational machine learning. ILP employs techniques of both Machine Learning and Logic Programming to build first-order logic theories from examples and background knowledge, which are also represented by means of first-order logic clauses. It allows the efficient representation of substantial knowledge about the problem, and allows this knowledge to be used during the learning process. The general idea underlying ILP is:

Given:

- a set of positive and negative examples $\boldsymbol{E} = \boldsymbol{E}^+ \cup \boldsymbol{E}^-$
- a predicate $\boldsymbol{p}$ specifying the target relation to be learned

- knowledge $\boldsymbol{K}$ of a certain domain, described according to a language $\boldsymbol{L_k}$, which specifies which other predicates $\boldsymbol{q_i}$ can be part of the definition of $\boldsymbol{p}$.

The goal is: to induce a hypothesis (or theory) $\boldsymbol{h}$ for $\boldsymbol{p}$, with relation to $\boldsymbol{E}$ and $\boldsymbol{K}$, which covers most of the $\boldsymbol{E}^+$, without covering the $\boldsymbol{E}^-$, that is, $\boldsymbol{K} \wedge \boldsymbol{h} \vDash \boldsymbol{E}^+$ and $\boldsymbol{K} \wedge \boldsymbol{h} \nvDash \boldsymbol{E}^-$.

To implement our approach we chose Aleph (Srinivasan, 2000), an ILP system which provides a complete relational learning inference engine and various customization options. We used the following options, which correspond to the Progol mode (Muggleton, 1995): bottom-up search, non-incremental and non-interactive learning, and learning based only on positive examples. Fundamentally, the default inference engine induces a theory iteratively by means of the following steps:

1. One instance is randomly selected to be generalized.

2. A more specific clause (bottom clause) explaining the selected example is built. It consists of the representation of all knowledge about that example.

3. A clause that is more generic than the bottom clause is searched, by means of search and generalization strategies (best first search, e.g.).

4. The best clause found is added to the theory and the examples covered by such clause are removed from the sample set. If there are more instances in the sample set, return to step 1.

3.3 Knowledge sources

The choice, acquisition, and representation of syntactic, semantic, and pragmatic knowledge sources (KSs) were our main concerns at this stage. The general architecture of the system, showing our 7 groups of KSs, is illustrated in Figure 2.

Several of our KSs have been traditionally employed in monolingual WSD (e.g., Agirre and Stevenson, 2006), while other are specific for MT. Some of them were extracted from our sample corpus (Section 3.1), while others were automatically extracted from lexical resources[1]. In what follows, we briefly describe, give the generic definition and examples of each KS, taking sentence (1), for the "to come", as example.

(1) "If there is such a thing as reincarnation, I would not mind *coming* back as a squirrel".

KS$_1$: Bag-of-words – a list of ±5 words (lemmas) surrounding the verb for every sentence (*sent_id*).

[1] Michaelis® and Password® English-Portuguese Dictionaries, LDOCE (Procter, 1978), and WordNet (Miller, 1990).

```
bag(sent_id, list_of_words).
bag(sent1,[mind, not, will, i, reincarnation, back, as, a,
squirrel])
```

KS$_2$: Part-of-speech (POS) tags of content words in a ±5 word window surrounding the verb.

```
has_pos(sent_id, word_position, pos).
has_pos(sent1, first_content_word_left, nn).
has_pos(sent1, second_content_word_left, vbp).
```

KS$_3$: Subject and object syntactic relations with respect to the verb under consideration.

```
has_rel(sent_id, subject_word, object_word).
has_rel(sent1, i, nil).
```

KS$_4$: Context words represented by 11 collocations with respect to the verb: 1st preposition to the right, 1st and 2nd words to the left and right, 1st noun, 1st adjective, and 1st verb to the left and right.

```
has_collocation(sent_id, collocation_type, collocation)
has_collocation(sent1, word_right_1, back).
has_collocation(sent1, word_left_1, mind).     …
```

KS$_5$: Selectional restrictions of verbs and semantic features of their arguments, given by LDOCE. Verb restrictions are expressed by lists of semantic features required for their subject and object, while these arguments are represented with their features.

```
rest(verb, subj_restrition, obj_ restriction ,translation)
rest(come, [], nil, voltar).
rest(come, [animal,human], nil, vir).          ...
```

```
feature(noun, sense_id, features).
feature(reincarnation, 0_1, [abstract]).
feature(squirrel, 0_0, [animal]).
```

The hierarchy for LDOCE feature types defined by Bruce and Guthrie (1992) is used to account for restrictions established by the verb for features that are more generic than the features describing the words in the subject / object roles in the sentence.

Ontological relations extracted from WordNet (Miller, 1990) are also used: if the restrictions imposed by the verb are not part of the description of its arguments, synonyms or hypernyms of those arguments that meet the restrictions are considered.

```
relation(word1, sense_id1, word2 ,sense_id2).
hyper(reincarnation, 1, avatar, 1).
synon(rebirth, 2, reincarnation, -1).
```

KS$_6$: Idioms and phrasal verbs, indicating that the verb occurring in a given context could have a specific translation.

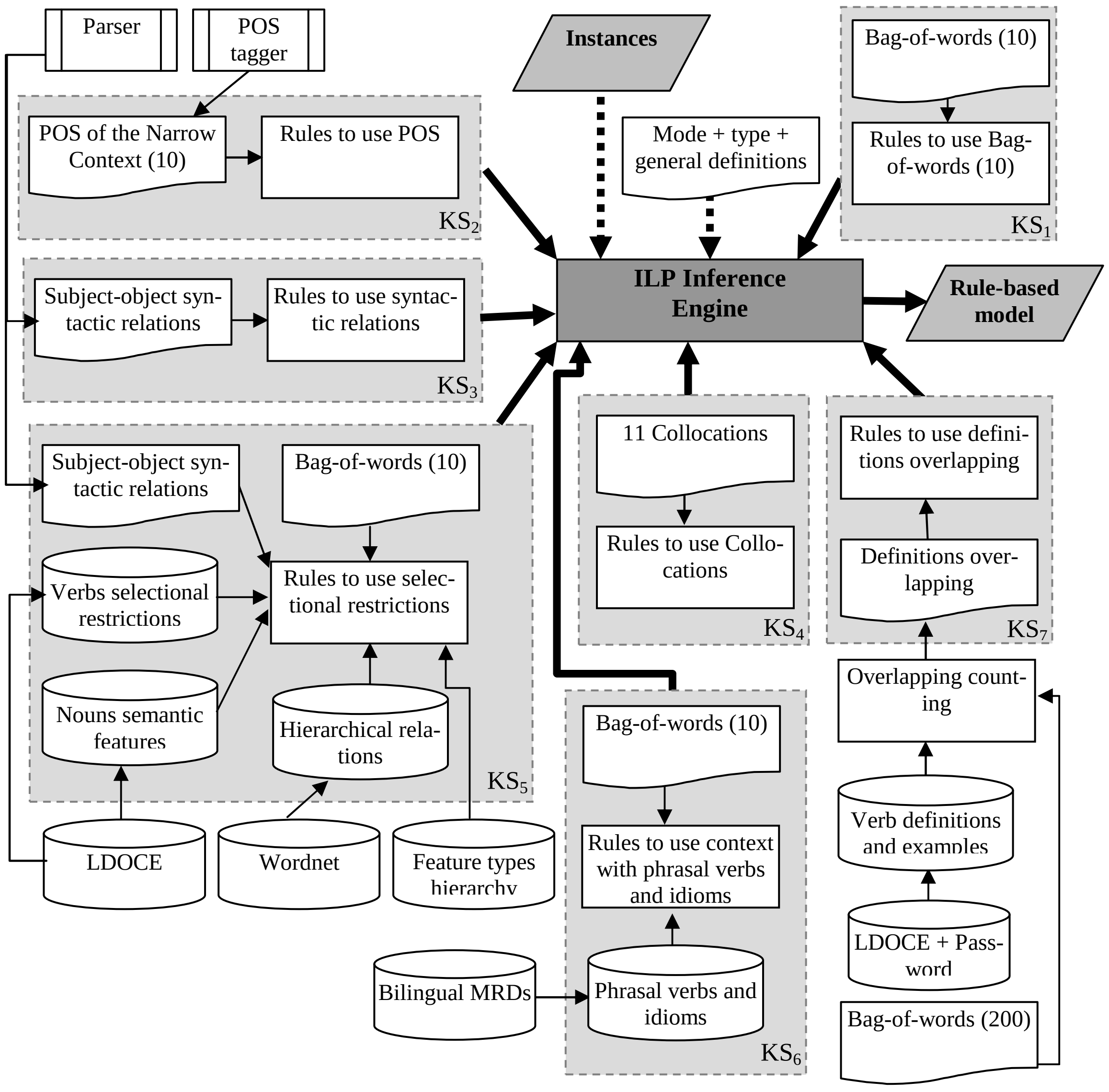

Parser
POS tagger
Instances
Bag-of-words (10)
Rules to use Bag-of-words (10)
KS1
POS of the Narrow Context (10)
Rules to use POS
KS2
Mode + type + general definitions
ILP Inference Engine
Rule-based model
Subject-object syntactic relations
Rules to use syntactic relations
KS3
Subject-object syntactic relations
Bag-of-words (10)
Verbs selectional restrictions
Rules to use selectional restrictions
Nouns semantic features
Hierarchical relations
KS5
11 Collocations
Rules to use Collocations
KS4
Rules to use definitions overlapping
Definitions overlapping
KS7
Overlapping counting
Verb definitions and examples
LDOCE + Password
Bag-of-words (200)
Bag-of-words (10)
Rules to use context with phrasal verbs and idioms
Phrasal verbs and idioms
KS6
Bilingual MRDs
LDOCE
Wordnet
Feature types hierarchy

Figure 2. System architecture

```
        exp(verbal_expression, translation)
exp('come about', acontecer).
exp('come about', chegar).                    …
```

KS$_7$: A count of the overlapping words in dictionary definitions for the possible translations of the verb and the words surrounding it in the sentence, relative to the total number of words.

```
   highest_overlap(sent_id, translation, overlapping).
highest_overlap(sent1, voltar, 0.222222).
highest_overlap(sent2, chegar, 0.0857143).
```

The representation of all KSs for each example is independent of the other examples. Therefore, the number of features can be different for different sentences, without resulting in sparseness in data.

In order to use the KSs, we created a set of rules for each KS. These rules are not dependent on particular words or instances. They can be very simple, as in the example shown below for bag-of-words, or more complex, e.g., for selectional restrictions. Therefore, KSs are represented by means of rules and facts (rules without conditions), which can be intensional, i.e., it can contain variables, making the representation more expressive.

```
has_bag(Sent,Word) :-
        bag(Sent,List), member(Word,List).
```

Besides the KSs, the other main input to the system is the set of examples. Since all knowledge about them is expressed by the KSs, the representation of examples is very simple, containing only the example identifier (of the sentence, in our case, such as, “sent1”), and the class of that example (in our case, the translation of the verb in that sentence).

```
            sense(sent_id,translation).
sense(sent1,voltar).
sense(sent2,ir).
```

In Aleph's default induction mode, the order of the training examples plays an important role. One example is taken at a time, according to its order in the training set, and a rule can be produced based on that example. Since examples covered by a certain rule are removed from the training set, certain examples will not be used to produce rules. Induction methods employing different strategies in which the order is irrelevant will be exploited in future work.

In order to produce a theory, Aleph also requires "mode definitions", i.e., the specification of the predicates ***p*** and ***q*** (Section 3.2). For example, the first mode definition below states that the predicate ***p*** to be learned will consist of a clause *sense(sent_id, translation)*, which can be instantiated only once (*1*). The other two definitions state the predicates ***q***, *has_colloc(sent_id, colloc_id, colloc)*, with at most *11* instantiations, and *has_bag(sent_id, word)*, with at most *10* instantiations. That is, the predicates in the conditional piece of the rules in the theory can consist of up to 11 collocations and a bag of up to 10 words. One mode definition must be created for each KS.

```
:- modeh(1,sense(sent,translation)).
:- modeb(11,has_colloc(sent,colloc_id,colloc)).
:- modeb(10,has_bag(sent,word)).    …
```

Based on the examples and background knowledge, the inference engine will produce a set of symbolic rules. Some of the rules induced for the verb "to come", e.g., are illustrated in the box below.

```
1. sense(A, sair) :-
   has_collocation(A, preposition_right, out).
2. sense(A, chegar) :-
   satisfy_restrictions(A, [animal,human],[concrete]);
   has_expression(A, 'come at').
3. sense(A, vir) :-
   satisfy_restriction(A, [human],[abstract]),
   has_collocation(A, word_right_1, from).
```

The first rule checks if the first preposition to the right of the verb is "out", assigning the translation "sair" if so. The second rule verifies if the subject-object arguments satisfy the verb restrictions, i.e, if the subject has the features "animal" or "human", and the object has the feature "concrete". Alternatively, it verifies if the sentence contains the phrasal verb "come at". Rule 3 also tests the verb selectional restrictions and the first word to the right of the verb.

4 Experiments and results

In order to assess the accuracy of our approach, we ran a set of initial experiments with our sample corpus. For each verb, we ran Aleph in the default mode, except for the following parameters:

set(evalfn, posonly): learns from positive examples.
set(search, heuristic): turns the search strategy heuristic.
set(minpos, 2): establishes as 2 the minimum number of positive examples covered by each rule in the theory.
set(gsamplesize, 1000): defines the number of randomly generated negative examples to prune the search space.

The accuracy was calculated by applying the rules to classify the new examples in the test set according to the order these rules appeared in the theory, eliminating the examples (correctly or incorrectly) covered by a certain rule from the test set. In order to cover 100% of the examples, we relied on the existence of a rule without conditions, which generally is induced by Aleph and points out to the most frequent translation in the training data. When this rule was not generated by Aleph, we add it to the end of theory. For all the verbs, however, this rule only classified a few examples (form 1 to 6).

In Table 2 we show the accuracy of the theory learned for each verb, as well as accuracy achieved by two propositional machine learning algorithms on the same data: Decision Trees (C4.5) and Support Vector Machine (SVM), all according to a 10-fold cross-validation strategy. Since it is rather impractical to represent certain KSs using attribute-value vectors, in the experiments with SVM and C4.5 only low level features were considered, corresponding to $\mathbf{KS_1}$, $\mathbf{KS_2}$, $\mathbf{KS_3}$, and $\mathbf{KS_4}$. On average, Our approach outperforms the two other algorithms. Moreover, its accuracy is by far better than the accuracy of the most frequent sense baseline (Table 1).

For all verbs, theories with a small number of rules were produced (from 19 to 33 rules). By looking at these rules, it becomes clear that all KSs are being explored by the ILP system and thus are potentially useful for the disambiguation of verbs.

5 Conclusion and future work

We presented a hybrid relational approach for WSD designed for MT. One important characteristic of our approach is that all the KSs were

Verb	Aleph Accuracy	C4.5 Accuracy	SVM Accuracy
come	0.82	0.55	0.6
Get	0.51	0.36	0.45
Give	0.96	0.88	0.88
Go	0.73	0.73	0.72
look	0.83	0.66	0.84
make	0.74	0.76	0.76
Take	0.66	0.35	0.41
Average	0.75	0.61	0.67

Table 2. Results of the experiments with Aleph

automatically extracted, either from the corpus or machine-readable lexical resources. Therefore, the work could be easily extended to other words and languages.

In future work we intend to carry out experiments with different settings: (a) combinations of certain KSs; (b) other sample corpora, of different sizes, genres / domains; and (c) different parameters in Aleph regarding search strategies, evaluation functions, etc. We also intend to compare our approach with other machine learning algorithms using all the KSs employed in Aleph, by pre-processing the KSs in order to extract binary features that can be represented by means of attribute-value vectors. After that, we intend to adapt our approach to evaluate it with standard WSD data sets, such as the ones used in Senseval[2].

References

E. Agirre and M. Stevenson. 2006 (to appear). Knowledge Sources for Word Sense Disambiguation. In *Word Sense Disambiguation: Algorithms, Applications and Trends,* Agirre, E. and Edmonds, P. (Eds.), Kluwer.

M.L. Brown, J.F. Kros. 2003. Data Mining and the Impact of Missing Data. *Industrial Management and Data Systems*, 103(8):611-621.

R. Bruce and L. Guthrie. 1992. Genus disambiguation: A study in weighted performance. In *Proceedings of the 14th COLING,* Nantes, pp. 1187-1191.

B.J. Dorr and M. Katsova. 1998. Lexical Selection for Cross-Language Applications: Combining LCS with WordNet. In *Proceedings of AMTA'1998,* Langhorne, pp. 438-447.

W.J. Hutchins and H.L. Somers. 1992. *An Introduction to Machine Translation.* Academic Press, Great Britain.

H. Lee. 2002. Classification Approach to Word Selection in Machine Translation. In *Proceedings of AMTA'2002,* Berlin, pp. 114-123.

G.A. Miller, R.T. Beckwith, C.D. Fellbaum, D. Gross, K. Miller. 1990. WordNet: An On-line Lexical Database. *International Journal of Lexicography,* 3(4):235-244.

R.J. Mooney. 1997. Inductive Logic Programming for Natural Language Processing. In *Proceedings of the 6th International ILP Workshop,* Berlin, pp. 3-24.

S. Muggleton. 1991. Inductive Logic Programming. *New Generation Computing,* 8 (4):295-318.

S. Muggleton. 1995. Inverse Entailment and Progol. *New Generation Computing Journal*, 13: 245-286.

B.S. Pedersen. 1997. *Lexical Ambiguity in Machine Translation: Expressing Regularities in the Polysemy of Danish Motion Verbs.* PhD Thesis, Center for Sprogteknologi, Copenhagen.

P. Procter (editor). 1978. *Longman Dictionary of Contemporary English.* Longman Group, Essex, England.

L. Specia. 2005. A Hybrid Model for Word Sense Disambiguation in English-Portuguese MT. In *Proceedings of the 8th CLUK,* Manchester, pp. 71-78.

L. Specia, M.G.V Nunes, M. Stevenson. 2005. Exploiting Parallel Texts to Produce a Multilingual Sense-tagged Corpus for Word Sense Disambiguation. In *Proceedings of RANLP-05,* Borovets, pp. 525-531.

A. Srinivasan. 2000. *The Aleph Manual. Technical Report.* Computing Laboratory, Oxford University. URL: http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph_toc.html.

M. Stevenson and Y. Wilks. 2001 The Interaction of Knowledge Sources for Word Sense Disambiguation. *Computational Linguistics*, 27(3):321-349.

D. Vickrey, L. Biewald, M. Teyssier, and D. Koller. 2005. Word-Sense Disambiguation for Machine Translation. In *Proceedings of HLT/EMNLP-05,* Vancouver.

N. Zinovjeva. 2000. *Learning Sense Disambiguation Rules for Machine Translation.* Master's Thesis, Department of Linguistics, Uppsala University.

[2] http://www.senseval.org/

On2L - A Framework for Incremental Ontology Learning in Spoken Dialog Systems

Berenike Loos
European Media Laboratory GmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
berenike.loos@eml-d.villa-bosch.de

Abstract

An open-domain spoken dialog system has to deal with the challenge of lacking lexical as well as conceptual knowledge. As the real world is constantly changing, it is not possible to store all necessary knowledge beforehand. Therefore, this knowledge has to be acquired during the run time of the system, with the help of the out-of-vocabulary information of a speech recognizer. As every word can have various meanings depending on the context in which it is uttered, additional context information is taken into account, when searching for the meaning of such a word.

In this paper, I will present the incremental ontology learning framework On2L. The defined tasks for the framework are: the hypernym extraction from Internet texts for unknown terms delivered by the speech recognizer; the mapping of those and their hypernyms into ontological concepts and instances; and the following integration of them into the system's ontology.

1 Introduction

A computer system, which has to understand and generate natural language, needs knowledge about the real world. As the manual modeling and maintenance of those knowledge structures, i.e. ontologies, are both time and cost consuming, there exists a demand to build and populate them automatically or at least semi automatically. This is possible by analyzing unstructured, semi-structured or fully structured data by various linguistic as well as statistical means and by converting the results into an ontological form.

In an open-domain spoken dialog system the automatic learning of ontological concepts and corresponding relations between them is essential, as a complete manual modeling of them is neither practicable nor feasible as the real world and its objects, models and processes are constantly changing and so are their denotations.

This work assumes that a viable approach to this challenging problem is to learn ontological concepts and relations relevant for a certain user - and only those - incrementally, i.e. at the time of the user's inquiry. Hypernyms[1] of terms that are not part of the speech recognizer lexicon, i.e. out-of-vocabulary (OOV) terms, and hence lacking any mapping to the employed knowledge representation of the language understanding component, should be found in texts from the Internet. That is the starting point of the proposed ontology learning framework *On2L* (On-line Ontology Learning). With the found hypernym On2L can assign the place in the system's ontology to add the unknown term.

So far the work described herein refers to the German language only. In a later step, the goal is to optimize it for English as well.

2 Natural Language and Ontology Learning

Before describing the actual ontology learning process it is important to make a clear distinction between the two fields involved: this is on the one hand natural language and on the other hand ontological knowledge.

As the Internet is a vast resource of up-to-date

[1]According to Lyons (1977) hyponymy is the relation which holds between a more specific lexeme (i.e. a hyponym) and a more general one (i.e. a hypernym). E.g. animal is a hypernym of cat.

information, On2L employs it to search for OOV terms and their corresponding hypernyms. The natural language texts are rich in terms, which can be used as labels of concepts in the ontology and rich in semantic relations, which can be used as ontological relations.

The two areas which are working on similar topics but are using different terminology need to be distinguished, so that the extraction of semantic information from natural language is separated from the process of integrating this knowledge into an ontology.

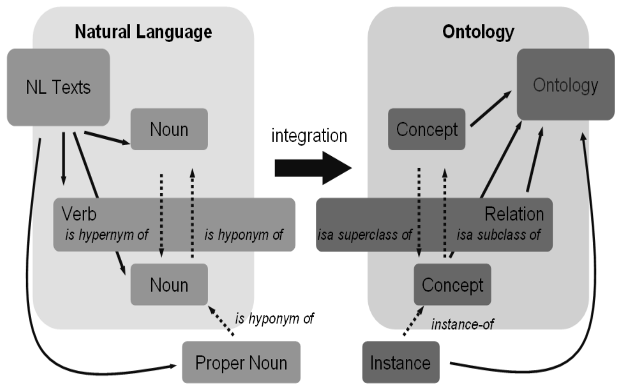


Figure 1: Natural Language and Ontology Learning

Figure 1 shows the process of ontology learning from natural language text. On the left side natural language lexemes are extracted. During a transformation process nouns, verbs and proper nouns are converted into concepts, relations and instances of an ontology[2].

3 Related Work

The idea of acquiring knowledge exactly at the time it is needed is new and became extremely useful with the emergence of open-domain dialog systems. Before that, more or less complete ontologies could be modeled for the few domains covered by a dialog system. Nonetheless, many ontology learning frameworks exist, which alleviate the work of an ontology engineer to construct knowledge manually, e.g. ASIUM (Faure and Nedellec, 1999), which helps an expert in acquiring knowledge from technical text using syntactic analysis for the extraction, a semantic similarity measure and a clustering algorithm for the conceptualization. OntoLearn (Missikoff et al., 2002) uses specialized web site texts as a corpus to extract terminology, which is filtered by statistical techniques and then used to create a domain concept forest with the help of a semantic interpretation and the detection of taxonomic and similarity relations. KAON Text-To-Onto (Maedche and Staab, 2004) applies text mining algorithms for English and German texts to semi-automatically create an ontology, which includes algorithms for term extraction, for concept association extraction and for ontology pruning.

Pattern-based approaches to extract hyponym/hypernym relationships range from hand-crafted lexico-syntactic patterns (Hearst, 1992) to the automatic discovery of such patterns by e.g. a minimal edit distance algorithm (Pantel et al., 2004).

The SmartWeb Project into which On2L will be integrated as well, aims at constructing an open-domain spoken dialog system (Wahlster, 2004) and includes different techniques to learn ontological knowledge for the system's ontology. Those methods work offline and not at the time of the user's inquiry in contrast to On2L:

C-PANKOW (Cimiano et al., 2005) puts a named entity into several linguistic patterns that convey competing semantic meanings. The patterns, which can be matched most often on the web indicate the meaning of the named entity.

RelExt (Schutz and Buitelaar, 2005) automatically identifies highly relevant pairs of concepts connected by a relation over concepts from an existing ontology. It works by extracting verbs and their grammatical arguments from a domain-specific text collection and computing corresponding relations through a combination of linguistic and statistical processing.

4 The ontology learning framework

The task of the ontology learning framework On2L is to acquire knowledge at run time. As On2L will be integrated into the open-domain dialog system Smartweb (Wahlster, 2004), it will be not only useful for extending the ontology of the system, but to make the dialog more natural and therefore user-friendly.

Natural language utterances processed by an open-domain spoken dialog system may contain words or parts of words which are not recognized by the speech recognizer, as they are not contained

[2]In our definition of the term *ontology* not only concepts and relations are included but also instances of the real world.

in the recognizer lexicon. The words not contained are most likely not represented in the word-to-concept lexicon as well[3]. In the presented ontology learning framework On2L the corresponding concepts of those terms are subject to a search on the Internet. For instance, the unknown term *Auerstein* would be searched on the Internet (with the help of a search engine like Google). By applying natural language patterns and statistical methods possible hypernyms of the term can be extracted and the corresponding concept in the ontology of the complete dialog system can be found. This process is described in Section 4.5.

As a term often has more than one meaning depending on the context in which it is uttered, some information about this context is added for the search[4] as shown in Section 4.4.

Figure 2 shows the life cycle of the On2L framework. In the middle of the diagram the question example by a supposed user is: *How do I get to the Auerstein?* The lighter fields in the figure mark components of the dialog system, which are only utilized by On2L, whereas the darker fields are especially built to complete the ontology learning task.

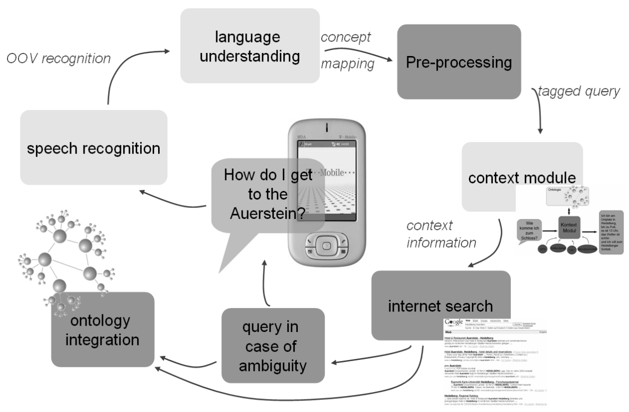


Figure 2: The On2L Life Cycle

The sequential steps shown in Figure 2 are described in more detail in the following paragraphs starting with the processing of the user's utterance by the speech recognizer.

4.1 Speech Recognition

The speech recognizer classifies all words of the user's utterance not found in the lexicon as out-of-vocabulary (OOV). That means an automatic speech recognition (ASR) system has to process words, which are not in the lexicon of the speech recognizer (Klakow et al., 2004). A solution for a phoneme-based recognition is the establishment of corresponding best rated grapheme-chain hypotheses (Gallwitz, 2002). These grapheme-chains are constructed with the help of statistical methods to predict the most likely grapheme order of a word, not found in the lexicon. Those chains are then used for a search on the Internet in the final version of On2L. To evaluate the framework itself adequately so far only a set of correctly written terms is subject to search.

4.2 Language Understanding

In this step of the dialog system, all correctly recognized terms of the user utterance are mapped to concepts with the help of a word-to-concept lexicon. Such a lexicon assigns corresponding natural language terms to all concepts of an ontology. This is not only a necessary step for the dialog system, but can assist the ontology learning framework in a possibly needed semantic disambiguation of the OOV term.

Furthermore the information of the concepts of the other terms of the utterance can help to evaluate results: when there are more than one concept proposal for an instance (i.e. on the linguistic side a proper noun like *Auerstein*) found in the system's ontology, the semantic distance between each proposed concept and the other concepts of the user's question can be calculated[5].

4.3 Preprocessing

A statistical part-of-speech tagging method decides on the most probable part-of-speech of the whole utterance with the help of the sentence context of the question. In the On2L framework we used the language independent tagger qtag[6], which we trained with the hand-tagged German corpus NEGRA 2[7].

[3]In case the speech recognizer of the system and the word-to-concept lexicon are consistent.

[4]Of course, even in the same context a term can have more than one meaning as discussed in Section 4.6.

[5]E.g. with the single-source shortest path algorithm of Dijkstra (Cormen et al., 2001).

[6]qtag exists as a downloadable JAR file and can therefore be integrated into a platform independent JAVA program. For more information, see http://www.english.bham.ac.uk/staff/omason/software/qtag.html (last access: 21st February 2006).

[7]The NEGRA corpus version 2 consists of 355,096 tokens (20,602 sentences) of German newspaper text, taken from the Frankfurter Rundschau. For more information visit: http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html (last access: 21st February 2006).

With the help of this information, the part-of-speech of the hypernym of the OVV term can be predicted. Furthermore, the verb(s) of the utterance can anticipate possible semantic relations for the concept or instance to be integrated into the ontology.

4.4 Context Module

To understand the user in an open-domain dialog system it is important to know the extra-linguistic context of the utterances. Therefore a context module is applied in the system, which can give information on the discourse domain, day and time, current weather conditions and location of the user. This information is important for On2L as well. Here we make use of the location of the user and the discourse domain so far, as this information is most fruitful for a more specific search on the Internet. The location is delivered by a GPS component and the discourse domain is detected with the help of the pragmatic ontology PrOnto ((Porzel et al., 2006)). Of course, the discourse domain can only be detected for domains modeled already in the knowledge base (Rueggenmann and Gurevych, 2004).

The next section will show the application of the context terms in more detail.

4.5 Hypernym extraction from the Internet

We apply the OOV term from the speech recognizer as well as a context term for the search of the most likely hypernym on the Internet.

For testing reasons a list of possible queries was generated. Here are some examples to give an idea:

(1) Auerstein – Heidelberg

(2) Michael Ballack – SportsDiscourse

(3) Lord of the Rings – CinemaDiscourse

On the left side of the examples 1 to 3 is the OOV term and on the right side the corresponding context term as generated by the context module. For searching, the part “Discourse” is pruned.

The reason to lay the main focus of the evaluation searches on proper nouns is, that those are most likely not in the recognizer lexicon and not as instances in the system’s ontology.

4.5.1 Global versus Local OOVs

To optimize results we make a distinction between global OOVs and local OOVs.

In the case of generally familiar proper nouns like stars, hotel chains or movies (so to say global OOVs), a search on Wikipedia can be quite successful.

In the case of proper nouns, only common in a certain country region, like Auerstein (Restaurant), Bierbrezel (Pub) and Lux (Cinema), which are local OOVs, a search with Wikipedia is generally not fruitful. Therefore it is searched with the help of the Google API.

As one can not know the kind of OOV beforehand, the Wikipedia search is started before the Google search. If no results are produced, the Google search will deliver them hopefully. If results are found, Google search will be used to test those.

4.5.2 Wikipedia Search

The structure of Wikipedia[8] entries is preassigned. That means, the program can know, where to find the most suitable information beforehand. In the case of finding hypernyms the first sentence in the encyclopedia description is most useful. To give an example, here is the first sentence for the search entry *Michael Ballack*:

(4) *Michael Ballack* (born September 26, 1976 in Grlitz, then East Germany) IS A German **football player**.

With the help of lexico-syntactic patterns, the hypernym can be extracted. Those so-called Hearst patterns (Hearst, 1992) occur frequently in lexicons for describing a term. In example 4 the pattern *X is a Y* would be matched and the hypernym *football player*[9] of the term *Michael Ballack* could be extracted.

4.5.3 Google Search

The search parameters in the Google API can be adjusted for the corresponding search task. The tasks we used for our framework are a search in the titles of the web pages and a search in the text of the web pages.

Adjusting the Google parameters The assumption was, that depending on the task the Google parameters should be adjusted. Four parameters were tested with the two tasks (Title and

[8]Wikipedia is a free encyclopedia, which is editable on the Internet: www.wikipedia.org (last access: 22nd February 2006)

[9]In German compounds generally consist of only one word, therefore it is easier to extract them than in the case of English ones.

Page Search, as described in the next paragraphs) and a combination thereof. The parameter *default* is used, when no other parameters are assigned; *intitle* is set, in case the search term should be found in the title of the returned pages; *allintext*, when the search term should be found in the text of the pages; and *inurl*, when the search term should be found in the URL.

In Figure 3 the outcome of the evaluation is shown. The evaluation was done by students, who scored the titles and pages with 1, when a possible hypernym could be found and 0 if not. Surprisingly, the default value delivered the best results for all tasks, followed by the allintext parameter.

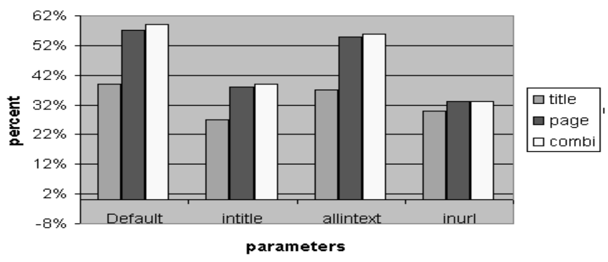


Figure 3: Evaluation of the Google parameters

Title Search To search only in the titles of the web pages has the advantage, that results can be generated relatively fast. This is important as time is a relevant factor in spoken dialog systems. As the titles often contain the hypernym but do not consist of a full sentence, Hearst patterns cannot be found. Therefore, an algorithm was implemented, which searches for nouns in the title, extracts them and counts the occurrences. The noun most frequently found in all the titles delivered by Google is regarded as the hypernym. For the counting we applied stemming and clustering algorithms to group similar terms.

Page Search For Page Search Hearst patterns as in Wikipedia Search were applied. In contrast to encyclopedia entries the recall of those patterns was not so high in the texts from the web pages.

Thus, we searched in the text surrounding of the searched term for nouns. Equally to Title Search we counted the occurrence of nouns. Different evaluation steps showed, that the window size of four words in front and after the term is most successful.

With the help of machine learning algorithms from the WEKA[10] library we did a text mining to ameliorate the results as shown in Faulhaber et al. (2006).

4.5.4 Results

Of all 100 evaluated pages for Google parameters only about 60 texts and about 40 titles contained possible hypernyms (as shown in Figure 3). This result is important for the evaluation of the task algorithms as well. The outcome of the evaluation setup was nearly the same: 38 % precicion for Title Search and about 58 % for Page Search (see Faulhaber (2006)). These scores where evaluated with the help of forms asking students: *Is X a hypernym of Y?*.

4.6 Disambiguation by the user

In some cases two or more hypernyms are scored with the same – or quite similar – weights. An obvious reason is, that the term in question has more than one meaning in the same context. Here, only a further inquiry to the user can help to disambiguate the OOV term. In the example from the beginning a question like “Did you mean the hotel or the restaurant?” could be posed. Even though the system would show the user that it did not perfectly understand him/her, the user might be more contributory than in a question like “What did you mean?”. The former question could be posed by a person familiar with the place, to disambiguate the question of someone in search for *Auerstein* as well and would therefore mirror a human-human dialog leading to more natural dialogs with the machine.

4.7 Integration into the ontology

The foundational ontology (Cimiano et al., 2004) integrated into the dialog system Smartweb is based on the highly axiomatized Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [11]. It features various extensions called *modules*, e.g. *Descriptions & Situations* (Gangemi and Mika, 2003). Additional to the foundational ontology a domain-independent layer is included which consists of a range of branches from the less axiomatic SUMO (Suggested Upper Merged Ontology (Niles and Pease, 2001)), which is known for its intuitive and comprehensible structure. Currently, the dialog system features several domain

[10]http://www.cs.waikato.ac.nz/ml/weka (last access: 21st February 2006).

[11]More information on this descriptive and reductionistic approach is found on the WonderWeb Project Homepage: wonderweb.semanticweb.org.

ontologies, i.e. a SportEvent-, a Navigation-, a WebCam-, a Media-, and a Discourse-Ontology.

According to this, it is possible that in some cases there exists the corresponding concept to a hypernym. This can be found out with the help of a so-called term widening. The concept labels in the SmartWeb Ontology are generally English terms. Therefore the found German hypernym has to be translated into English. An English thesaurus is used to increase the chance of finding the right label in the ontology.

5 Future Work

The work described here is still in process and not evaluated in detail so far. Therefore, our goal is to establish a task-oriented evaluation setup and to ameliorate the results with various techniques.

As natural language texts are not only rich in hierarchical relations but in other semantic relations as well, it is advantageous to extend the ontology by those relations.

As user contexts are an important part of a dialog system, we are planning to learn new user contexts, which can be represented in the ontology by the DOLCE module Descriptions and Situations.

Furthermore our goal is, to integrate the ontology learning framework into the open-domain spoken dialog system Smartweb.

References

Philipp Cimiano, Andreas Eberhart, Daniel Hitzler, Pascal Oberle, Steffen Staab, and Rudi Studer. 2004. The smartweb foundational ontology. *SmartWeb Project Report*.

Philipp Cimiano, Günter Ladwig, and Steffen Staab. 2005. Gimme' the context: Context-driven automatic semantic annotation with c-pankow. In *Proceedings of the 14th World Wide Web Conference*. ACM Press.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. Section 24.3: Dijkstra's algorithm. In *Introduction to Algorithms, Second Edition*, pages 595–601. MIT Press and McGraw-Hill.

Arndt Faulhaber, Berenike Loos, Robert Porzel, and Rainer Malaka. 2006. Towards understanding the unknown: Open-class named entity classification in multiple domains. In *Proceedings of the Ontolex Workshop at LREC*. Genoa, Italy.

David Faure and Claire Nedellec. 1999. Knowledge acquisition of predicate argument structures from technical texts using machine learning: The system asium. In *EKAW '99: Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management*, London, UK. Springer-Verlag.

Florian Gallwitz. 2002. *Integrated Stochastic Models for Spontaneous Speech Recognition*. Logos, Berlin.

Aldo Gangemi and Peter Mika. 2003. Understanding the semantic web through descriptions and situations. In *Proceedings of the ODBASE Conference*. Springer.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*, Nantes, France.

Dietrich Klakow, Georg Rose, and Xavier Aubert. 2004. Oov-detection in a large vocabulary system using automatically defined word-fragments as filler. In *Proceedings of EUROSPEECH'99*, Budapest, Hungary.

John Lyons. 1977. *Semantics*. University Press, Cambridge, MA.

Alexander Maedche and Steffen Staab. 2004. Ontology learning. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems. Springer.

Michele Missikoff, Roberto Navigli, and Paola Velardi. 2002. Integrated approach to web ontology learning and engineering. In *IEEE Computer - November*.

Ian Niles and Adam Pease. 2001. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, *Workshop on Ontology Management*, Ogunquit, Maine. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001).

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale semantic acquisition. In *Proceedings of Coling*, Geneva, Switzerland. COLING.

Robert Porzel, Hans-Peter Zorn, Berenike Loos, and Rainer Malaka. 2006. Towards a separation of pragmatic knowledge and contextual information. In *Proceedings of ECAI-06 Workshop on Contexts and Ontologies*, Lago di Garda, Italy.

Klaus Rueggenmann and Iryna Gurevych. 2004. Assigning domains to speech recognition hypotheses. In *Proceedings of HLT-NAACL Workshop on Spoken Language Understanding for Conversational Systems and Higher Level Linguistic Knowledge for Speech Processing*. Boston, USA.

Alexander Schutz and Paul Buitelaar. 2005. Relext: A tool for relation extraction in ontology extension. In *Proceedings of the 4th International Semantic Web Conference*. Galway, Ireland.

Wolfgang Wahlster. 2004. SmartWeb: Mobile applications of the semantic web. In *Proceedings of Informatik*, Ulm, Germany.

Focus to Emphasize Tone Structures for Prosodic Analysis in Spoken Language Generation

Lalita Narupiyakul
Faculty of Computer Science, Dalhousie University
6050 University Avenue, Halifax, Nova Scotia, Canada B3H 1W5
Tel. +1-902-494-6441, Fax. +1-902-494-3962
lalita@cs.dal.ca

Abstract

We analyze the concept of focus in speech and the relationship between focus and speech acts for prosodic generation. We determine how the speaker's utterances are influenced by speaker's intention. The relationship between speech acts and focus information is used to define which parts of the sentence serve as the focus parts. We propose the Focus to Emphasize Tones (FET) structure to analyze the focus components. We also design the FET grammar to analyze the intonation patterns and produce tone marks as a result of our analysis. We present a proof-of-the-concept working example to validate our proposal. More comprehensive evaluations are part of our current work.

1 Introduction

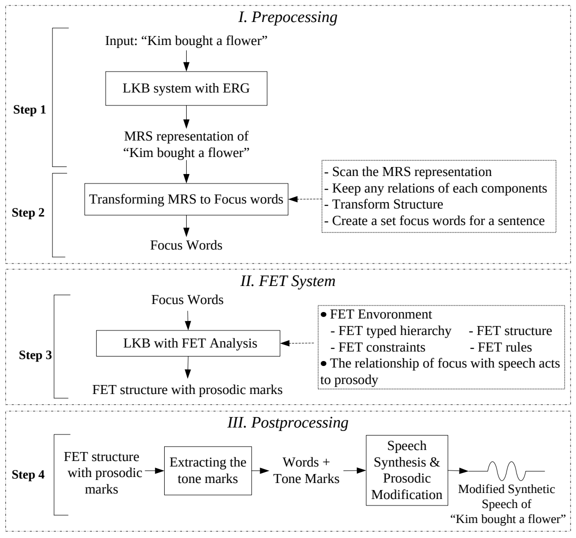
A speaker's utterance may convey different meaning to a hearer. Such ambiguities can be resolved by emphasizing accents in different positions. Focus information is needed to select correct positions for accent information. To determine focus information, a speaker's intentions must be revealed. We apply speech act theory to written sentences, our input, to determine a speaker's intention. Subsequently our system will produce a speaker utterance, the result of analysis.

Several research publications, such as (Steedman and Prevost, 1994) and (Klein, 2000), explore prosodic analysis for spoken language generation (SLG). Klein (2000) designs constraints for prosodic structures in the HPSG framework. His approach is based on an isomorphism of syntactic and prosodic trees. This approach is heavily syntax-driven and involves making prosodic trees by manipulation of the syntactic trees. This approach results in increased complexity since the type hierarchy of phrases must cross-classify prosodic phrases under syntactic phrases. Haji-Abdolhosseini (2003) extended Klein's approach. Rather than referring to syntax, Haji-Abdolhosseini sets the information domain to interact between the syntactic-semantic domain and the prosodic domain. His work reduces the complexity of type hierarchies and constraints which are not related to the syntactic structure. He designs the information structure and defines constraints for the HPSG framework. However his work limits the number of tone selections because he only defines two tone marks: rise-fall-rise and fall to annotate a sentence.

Our work is inspired by Haji-Abdolhosseini's work. We design the focus structure for spoken language generation. Based on the focus theory (Von Heusinger, 1999), the focus part identifies what part of the sentence can be marked with the strong accent or emphasized by a high tone. By analyzing speech acts, we can understand how speech with prosody can convey distinct speaker intentions to a hearer. In the next section, we present an overview of our FET (Focus to Emphasize Tone) system and its processes. We will explain how to analyze focus information, design the FET structure, and find the relationships of focus with speech acts to prosodic marks in section 3. We implement our FET grammar for the Linguistic Knowledge Base (LKB) system (Copestake, 2002), generate a set of focus words, explain the FET environment, and show an example in section 4. In the last section, we conclude the current state of our work and the future work.

2 Overview of FET System for Prosodic Analysis in SLG

Our system generates the prosodic structure depending on the focus analysis. We use this prosodic structure to modify synthetic speech for SLG. Our FET structure is constrained by the speaker's intention. To define prosody, we explore the relationships of focus and speech acts from various sentence types. The diagram of our FET system is shown in figure 1 and we present an overview of the FET system based on the LKB system below.



I. Prepocessing
Input: "Kim bought a flower"
LKB system with ERG
Step 1
MRS representation of "Kim bought a flower"
Transforming MRS to Focus words
- Scan the MRS representation
- Keep any relations of each components
- Transform Structure
- Create a set focus words for a sentence
Step 2
Focus Words
II. FET System
Focus Words
LKB with FET Analysis
• FET Envoronment
- FET typed hierarchy - FET structure
- FET constraints - FET rules
• The relationship of focus with speech acts to prosody
Step 3
FET structure with prosodic marks
III. Postprocessing
Step 4
FET structure with prosodic marks
Extracting the tone marks
Words + Tone Marks
Speech Synthesis & Prosodic Modification
Modified Synthetic Speech of "Kim bought a flower"

Figure 1: A diagram of the FET system

Our input is a sentence and its focus criterion obtained from a user. In figure 1, the example sentence is "Kim bought a flower" and the focus criterion is G (see table 2). Our system is composed of four main steps.

The first step is preprocessing. The LKB system with the English Resource Grammar (ERG) (Copestake, 2002) parses a sentence. The LKB system analyzes the syntactic and semantic structures and generates the Minimal Recursive Semantic (MRS) (Copestake et al., 1995) representation. This step occurs before invoking the FET system.

In the second step, we scan the MRS structure and collect any components and their relations among them obtained from the preprocessing step. We select only required information, such as sentence mood, from the MRS representation, assign a speech act code referring to a main verb of a sentence, and obtain from the MRS structure a set of focus words. These focus words are an input for the focus information analysis in the FET system.

The third step is the FET analysis. This step generates the prosodic components inside the FET structure. Using our FET grammar, we input the focus words into the LKB system with the FET environment. This environment consists of the FET type hierarchy, constraints, rules, and structures including the focus and prosodic features. Since the LKB system with FET environment can analyze the focus relations corresponding to speech acts and sentence moods, the system completes the FET structure by generating a set of appropriate prosodic structures containing prosodic marks as a result.

The last step is the postprocessing process. We extract words and their prosodic marks as Tone and Break Index (ToBI) representations (Silverman et al., 1992) from the FET structure. The extracting system processes the FET structure, extracts only our required prosodic fields. These fields are a set of words and their tone marks for a sentence. We use the set of words with tone marks to modify synthetic speech, which is generated by speech synthesis. We use the PRAAT (Boersma and Weenink, 2005) to modify the prosody of the synthetic speech for a sentence. Our output is an audio file of the sentence with modified prosody. Modifying prosody follows the tone marks which are analyzed by the FET system.

3 FET Analysis

We describe our concept of the FET analysis (see step 3, figure 1). We determine how the speaker's utterances are influenced by a speaker's intention. Focus information can be used to indicate how to appropriately mark a part of a sentence to convey the speaker's intention. Focus can scope the content in a sentence to which a speaker wants the listener to pay attention. We also consider speech acts which involve a speaker's intention and speaker's utterance. We analyze the relationships of focus parts with speech acts to tone marks. We define the intonation patterns depending on particular focus parts and speech acts. Our FET analysis obtains syntactic and semantic contents from the preprocessing process. We employ the LKB system to parse a sentence. The LKB system is an HPSG parser. A particular grammar, used for LKB system, is called ERG containing more than 10,000 lexemes. The LKB system generates the semantic information which is represented by MRS representation.

3.1 FET Constraints

Our FET analysis uses a constraint-based approach. We find what part (actor, act, actee or their combinations) must be in the focus from the the MRS structure. If the focus is marked at a position in a sentence then the speaker wants the hearer to recognize the content at that position in the sentence. For example, the speaker utters the sentence "Kim bought a flower" by emphasizing at the different positions in the sentence as shown table 1. Then we transform the MRS structures to our FET content structure which is represented by a set of focus words. This structure contains "actor" (a person or a thing that acts something in a sentence), "act" (an activity in that sentence), and "actee" (the response of the activity) parts.

Table 1: The different focuses in the sentence

	Focus	**Speaker wants to focus at . . .**
$[a]$	$[KIM]_F$ bought a flower.	(Who bought a flower?)
$[b]$	Kim bought [a FLOWER]$_F$.	(What did Kim buy?)
$[c]$	Kim [BOUGHT a flower]$_F$.	(What did Kim do?)

Considering a focus part, our focus model will acknowledge two focus types: *w-focus*, and *s-focus*. The *w-focus* represents wide focus, which covers a phrase or a word. The *s-focus* represents single focus, which is placed on a word in the sentence. We assign the actor and actee parts as single or wide focus while the act part is only an *s-focus*. Normally, the focus does not cover only the act part. If the focus covers the act part, then the focus must cover at least one of the related parts (actor or actee). Therefore, we set the focus types following all situations that occur and call the focus criteria. Eight focus criteria are shown in table 2.

Table 2: The focus parts and the focus types

No.	Focus Parts	Focus Types
A	actor+act+actee	{w-focus(actor),s-focus(act),w-focus(actee)} or undefined
B	actor+act	{w-focus(actor),s-focus(act)}
C	actor+actee	{w-focus(actor),s-focus(actee)} or {w-focus(actee),s-focus(actor)}
D	actor	w-focus(actor) or s-focus(actor)
E	act+actee	{s-focus(act),w-focus(actee)}
F	act	s-focus(act)
G	actee	w-focus(actee) or s-focus(actee)
H	$\oslash$	undefined

We define constraints to select the focus types following the different situations. We categorize the conditions for focus types to five cases. These conditions cover all possible situations. These situations define the focus based on the focus parts for most simple sentences. We illustrate the attribute value matrix (AVM) structure to represent these situations in figure 2.

(a) *An s-focus of the actor or actee parts.* The last node in the list of objects is defined as the focus position to emphasize tone (*FET-obj*), see figure 2(a).

(b) *A w-focus at the actor or actee parts.* The list of objects is the FET-obj in the sentence as shown in figure 2(b).

(c) *A w-focus at actor or actee parts containing the multiple lists of objects.* The lists are merged together to be the *FET-obj* as shown in figure 2(c).

(d) *An s-focus at actor or actee parts containing the multiple lists of objects.* If the focus type is an *s-focus* and there are *m* sets of lists of objects (multiple lists of objects), then these lists of objects can be split into the *s-focus* of each list of objects, see figure 2(d).

(e) *A focus on the act part.* Two cases of defining the focus types are shown in figure 2(e). The first case, the *s-focus* marks the act part while the *w-focus* marks the actee part. The second case, the *s-focus* marks the act part and the *w-focus* marks at the actor part.

$$make\_s-focus \rightarrow focus-struc\,\& \begin{bmatrix} Focus-Type & s-focus \\ list-focus & <a_1,a_2,\ldots,a_n> \\ FET-obj & <a_n> \end{bmatrix} \qquad make\_w-focus \rightarrow focus-struc\,\& \begin{bmatrix} Focus-Type & w-focus \\ list-focus & <a_1,a_2,\ldots,a_n> \\ FET-obj & <a_1,a_2,\ldots,a_n> \end{bmatrix}$$

(a) (b)

$$merg\_list\_w-focus \rightarrow focus-struc\,\& \begin{bmatrix} Focus-Type & w-focus \\ list-focus & <[a_1,a_2,\ldots,a_n],..,[m_1,m_2,\ldots,m_n]> \\ FET-obj & <a_1,a_2,\ldots,a_n,...,m_1,m_2,\ldots,m_n> \end{bmatrix}$$

(c)

$$split\_list\_s-focus \rightarrow focus-struc\,\& \begin{bmatrix} Focus-Type & s-focus \\ list-focus & <a_1,a_2,\ldots,a_n> \\ FET-obj & <a_n> \end{bmatrix} \vee\ldots\vee\; focus-struc\,\& \begin{bmatrix} Focus-Type & s-focus \\ list-focus & <m_1,m_2,\ldots,m_n> \\ FET-obj & <m_n> \end{bmatrix}$$

(d)

$$make\_act\_s-focus :\rightarrow \left\{ focus-struc\,\& \begin{bmatrix} act \\ Focus-Type & s-focus \\ list-focus & <a_1,a_2,...,a_n> \\ FET-obj & <a_n> \end{bmatrix}, focus-struc\,\& \begin{bmatrix} actee \\ Focus-Type & w-focus \\ list-focus & <b_1,b_2,...,b_n> \\ FET-obj & <b_1,b_2,...,b_n> \end{bmatrix} \right\} \vee \left\{ focus-struc\,\& \begin{bmatrix} actor \\ Focus-Type & w-focus \\ list-focus & <c_1,c_2,...,c_n> \\ FET-obj & <c_1,c_2,...,c_n> \end{bmatrix}, focus-struc\,\& \begin{bmatrix} act \\ Focus-Type & s-focus \\ list-focus & <a_1,a_2,...,a_n> \\ FET-obj & <a_n> \end{bmatrix} \right\}$$

(e)

Figure 2: The AVM structure of focus marking: For actor or actee part, (a) *s-focus* (b) *w-focus* (c) *w-focus* of the multiple lists (d) *s-focus* of the multiple lists and, (e) *s-focus* for act part

3.2 The Relationships of Focus with Speech Acts to Prosody

At step 3 of figure 1, we define the speech act codes following Brennenstuhl (1981). To mark these codes, we consider the main verb (known as the act part inside the FET content structure). These codes define what the speech act categories can be in each sentence. A sentence can be marked by more than one code according to speech act classification (Ballmer and Brennenstuhl, 1981). We mark the speech act codes for 62 sentences from a part of the CMU communicator dataset (2002). Considering the relationships between speech acts and focus parts, we found some common patterns for marking tones in a sentence. For example, the tone mark L-L%, analyzed as low phrase tone (L-) to low boundary tone (L%), is marked at the last word of a sentence for any affirmative sentence. The tone marks H- (high phrase tone) and L- are marked at the last word before conjunction (such as "and", "or", "but", and so on) or are marked at the last word of the current phrase (following the next phrase). We know that the tone mark H* (high accent tone) is used to emphasize a word or a group of words in a sentence. If we want strong emphasis at a word or a group of words then we use the tone mark L+H* (rising accent tone) instead of H*. The groups of speech acts, that we consider in this paper, include intending (EN0ab), want (DE8b), and victory (KA4a), to explore tone patterns. We analyze the relationships of speech acts and tone marks grouping by focus parts as shown in figure 3. Since our example sentence has focus at actee part, speech act code is en0ab, and the sentence mood is affirmative sentence (aff), we define the tone marks for a set of words in the actee part as L+H* L-L%, following figure 3. The outcome of this process is the FET structure including the prosodic structure.

Code	Act Type	Sent Type	Condition
EN0ab	Actee	Aff	$Actee\_tone \leftarrow ([L* \vee (L+H*)] + L\text{-})^{n-1} + ([L* \vee (L+H*)] + L\text{-}L\%)$
		Int	$Actee\_tone \leftarrow ([H* \vee (L+H*)] + H\text{-})^{n-1} + ([H* \vee (L+H*)] + H\text{-}H\%)$
DE8b	Actee	Aff	$Actee\_tone \leftarrow H* + L\text{-}L\%$
		Int	$Actee\_tone \leftarrow H* + H\text{-}H\%$
KA4a	Actor	Aff	$Actor\_tone \leftarrow H* \vee (L+H*)$
		Int	$Actor\_tone \leftarrow H* \vee (L+H*)$
	Actee	Aff	$Actee\_tone \leftarrow ([H* \vee (L+H*)] + L\text{-})^{n-1} + L\text{-}L\%$
		Int	$Actee\_tone \leftarrow ([H* \vee (L+H*)] + H\text{-})^{n-1} + H\text{-}H\%$

Figure 3: Tone constraints

4 An Example of FET Implementation with LKB System

In this section, we implement our system using the LKB system with the FET environment. We analyze an example sentence "Kim bought a flower" using the FET system. The system contains the FET environment (see section 4.2) and constrains focus and prosodic features based on FET analysis in section 3. We introduce the FET type hierarchy and describe the components of FET structure.

4.1 Interpreting the MRS representation for Focus Words

In the preprocessing process, the LKB system with ERG parses a sentence and generates the MRS representation (see step 1, figure 1). By scanning each object inside the MRS representation, we keep all reference numbers, mapped with their objects and record every connection that is related to this object and this reference number. We extract only necessary information to generate a set of focus words (see step 2, figure 1). These focus words are generated to correspond to the LKB system. For a sentence, we define a speech act code referring to a main verb and obtain a focus criterion from a user.

Each focus word, as shown in figure 4, is marked by a focus part (*focus-part*). A focus word structure (*focus-word*) contains the focus criterion (*fcgroup*), speech act code (*spcode*), sentence mood (*stmood*) and focus position (*focus-pos*) in a focus part. In figure 4, the focus criterion is defined as group G (see table 2) while the speech acts code is en0ab (intending). The sentence mood referring from MRS is affirmative sentence and focus position is the last node (*ls*). We will describe the *focus-word* and its components in the next section. In figure 4, "Kim" is a actor part while "bought" is an act part. The words "a" and "flower" are the actee parts.

```
Kim := focus-word &                   bought := focus-word &
[ ORTH "Kim",                         [ ORTH "bought",
  HEAD actor-part &                     HEAD act-part &  [ AGR1 ls-act_G-aff-en0ab ],
       [ AGR1 ls-actor_G-aff-en0ab],    SPR < [HEAD actor-part &
  SPR <  >,                                   [ AGR1 ls-actor_G-aff-en0ab ] ] >,
  COMPS < > ].                          COMPS < focus-phrase & [HEAD actee-part &
                                              [AGR1 ls-actee_G-aff-en0ab  ]]  > ].
a := focus-word &                     flower := focus-word &
[ ORTH "a",                           [ ORTH "flower",
  HEAD actee-part &                     HEAD actee-part &
       [AGR1 pv-actee_G-aff-en0ab ],          [AGR1 ls-actee_G-aff-en0ab  ],
  SPR < >,                              SPR < [ HEAD actee-part &
  COMPS < > ].                                [AGR1 pv-actee_G-aff-en0ab ]] >,
                                        COMPS < focus-phrase & [HEAD actee-part &
                                              [AGR1 ls-actee_G-aff-en0ab  ]]> ].
```

Figure 4: A set of focus words

4.2 FET Tone Environment

In FTE system, we provide a set of focus words to the LKB system with the FET environment (see step 3, figure 1). This environment contains the constraints, rules, type hierarchy, a set of features, and their structures for the FET analysis. We design the FET type hierarchy as shown in figure 5. We define three main groups of feature structures: **focus-value**, **prosodic-value** and *feat-struc* to control the focus constraints. **focus-*

*value** represents the focus structures. It is composed of five subfeature structures: focus criterion, focus type (*fctype*), focus name (*focus*), focus position (*focus-pos*), and checking whether a tone mark can be marked at a word (tone-mark). **prosody-value** represents the prosodic structure. Four prosodic subfeature structures are sentence mood, speech act code, accent tone (*accent-tone*), and boundary tone (*bound-tone*). *feat-struc* contains the core FET structure that constrains the relationships between focus and prosodic features. The *feat-struc* structure is composed of six main subfeature structures: (i) focus category structure (*focus-cat*) is a set of constraints which are the combinations of a focus part and a focus criterion such as act_g, actor_g, actee_g, and so on, (ii) focus part structure (*focus-part*) classifies act part and non-act part as actor part or actee part, (iii) focus structure (*focus-struc*) is a subfeature structure of *focus-word* and *focus-phrase*, (iv) checking whether prosodic marks can be marked (*prosody*), (v) prosodic mark (*prosody-mark*) structure maps between types of prosodic mark and accent and boundary tones: no-mark, hEm_Sh-break, etc, (vi) a set of prosodic marks (*prosody-set*) is a set of combinations between accent and boundary tones.

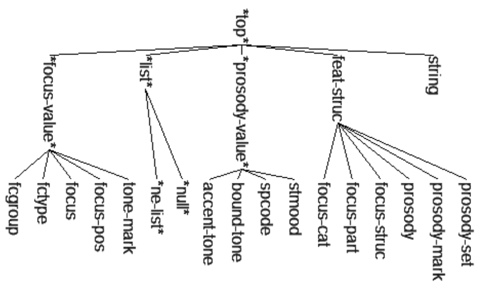


Figure 5: FET type hierarchy

4.2.1 Focus Structure

In figure 6(a), the *focus-phrase* inherits the *focus-struc* with a feature ARGS. The ARGS represents a list of words in the sentence. The focus rules parse the *focus-phrase* with their constraints and define whether tone can be marked at a word in each focus part. The *focus-word* inherits the *focus-struc* with orthography of a word (ORTH) as string. The *focus-word*, as shown in 6(b), represents the focus content structure and corresponds to the LKB system. The *focus-struc*, as show in figure 6(c), consists of HEAD, specifier (SPR) and complement (COMPS) (Ivan et al., 2003). Inside the *focus-struc*, HEAD refers to *focus-part* which is shown in figure 6(d). SPR and COMP are used to specify the components of previous nodes and following nodes in a sentence. Each *focus-part* contains focus and prosodic structures. We classify focus following the possible *focus-cat* for the FET structure. The *focus-cat* controls the constraints for the actor, act and actee parts. The *focus-cat* contains both the focus and prosodic features as a set of subfeatures of the FET structure. This structure contains focus position, focus group, focus type, a set of prosody marks and prosodic structure (*prosody*). The *focus-cat* is shown in figure 6(e).

$$
\begin{array}{l} focus-phrase := \\ focus-struc\ \& \\ \begin{bmatrix} ARGS & *list* \end{bmatrix} \end{array}
\qquad
\begin{array}{l} focus-word := \\ focus-struc\ \& \\ \begin{bmatrix} ORTH & string \end{bmatrix} \end{array}
\qquad
\begin{array}{l} focus-struc := \\ feat-struc\ \& \\ \begin{bmatrix} HEAD & focus-part \\ SPR & *list* \\ COMPS & *list* \end{bmatrix} \end{array}
$$

(a) (b) (c)

$$
\begin{array}{l} focus-part := \\ feat-struc\ \& \\ \begin{bmatrix} FOCUS & focus \\ AGR1 & focus-cat \end{bmatrix} \end{array}
\qquad
\begin{array}{l} focus-cat := \\ feat-struc\ \& \\ \begin{bmatrix} FOCUS-POS & focus-pos \\ FCGROUP & fcgroup \\ FCTYPE & fctype \\ ADDTONE & addtone \\ PROSODY & prosody \end{bmatrix} \end{array}
$$

(d) (e)

Figure 6: Type feature structure of: (a) *focus-phrase* (b) *focus-word* (c) *focus-struc* (d) *focus-part* (e) *focus-cat*

4.2.2 Prosodic Structure

The prosodic structure consists of these subfeatures: sentence mood, speech act code, and a set of prosodic mark structures. This structure controls the prosodic marks following the FET constraints. These constraints depend on the relationships of focus with speech acts to intonation patterns. The prosody structure is shown in figure 7(a). The accent and boundary tones are mapped with the *prosody-mark* which is illustrated in figure 7(b).

$$
\begin{array}{l} prosody := \\ feat-struc\ \& \\ \begin{bmatrix} STMOOD & stmood \\ SPCODE & spcode \\ PROSODY-MARK1 & prosody-mark \\ PROSODY-MARK2 & prosody-mark \end{bmatrix} \end{array}
\qquad
\begin{array}{l} prosody-mark := \\ feat-struc\ \& \\ \begin{bmatrix} ACCENT-TONE & accent-tone \\ BOUND-TONE & bound-tone \end{bmatrix} \end{array}
$$

(a) (b)

Figure 7: Type feature structure of: (a) Prosodic structure (b) Prosodic mark structure

For focus rules, we have two types of focus rules that are head-complement and head-specifier rules. These rules process the same as a simple grammar rule which is explained in (Ivan et al., 2003). Using these rules, the example sentence "Kim bought a flower" is parsed and the result is the complete FET structure including the focus and prosodic information. The FET structure of the word “Kim” is shown in figure 8.

```
[ focus-word
  HEAD: [ actor-part
          FOCUS: ACTOR
          AGR1: [ ls-actor\_g-aff-en0ab
                FOCUS-POS: LAST
                FCGROUP: G
                FCTYPE: WS-FOCUS
                PROSODY: [ aff-en0ab-no-prosody
                         STMOOD: AFF
                         SPCODE: EN0AB
                         PROSODY-SET: [ no-prosody
                             PROSODY-MARK1: [ no-mark
                                 ACCENT-TONE: NOACCENT
                                 BOUND-TONE: NOBOUND ] ]
                             PROSODY-MARK2: [ no-mark
                                 ACCENT-TONE: NOACCENT
                                 BOUND-TONE: NOBOUND ]
                           ] ] ] ] ] ] ] ]
  SPR: *NULL*
  COMPS: *NULL*
  ORTH: Kim ]
```

Figure 8: FET structure of the word “Kim”

4.3 Modifying Prosody for Synthetic Speech

In the postprocessing process (see step 4, figure 1), we extract a set of words with tone marks from the FET structure. An example of these words with tone marks is shown in figure 9. Finally we transfer this data to generate the synthesized speech by the speech synthesis and modify prosody.

```
ORTH: Kim                 ORTH: a
Focus: actor-part         Focus: actee-part
ACCENT_TONE1: NOACCENT    ACCENT_TONE1: NOACCENT
BOUND_TONE1: NOBOUND      BOUND_TONE1: NOBOUND

ORTH: bought              ORTH: flower
Focus: act-part           Focus: actee-part
ACCENT_TONE1: NOACCENT    ACCENT_TONE1: L+H*
BOUND_TONE1: NOBOUND      BOUND_TONE1: L-L%
```

Figure 9: A set of words and their tone marks

5 Concluding Remarks

We design the FET system based on the small number of sentences from a part of the CMU communicator dataset (2002). These simple sentences relate to traveling information. In this paper, we use the MRS representation from the LKB system to determine actor, act and actee parts. Since the LKB has a limited grammar and produces multiple parses, then we assume that our input sentence can be parsed by the HPSG parser and only a correct output is provided to the LKB system with the FET environment. We analyze the relationships of focus with speech acts to tone marks. To mark tone, we group the tone patterns by speech acts and focus parts. We implement the FET system using LKB and an example is illustrated in section 4 in this paper. Using the LKB with the FET grammar, the system can parse most simple sentences from the CMU communicator dataset and generate the complete FET structure including prosodic marks for each sentence. We are evaluating the FET system with respect to three aspects: appreciation of listeners to tone based on the tone marks from the FET system, conveying the focus content in a sentence to listeners and the correctness of prosodic annotation. In the future, we will finish the evaluations and analyze more relationships of focus with speech acts to tones to support the various sentences.

Acknowledgement

This work is supported by NSERC, Canada, Royal Golden Jubilee Ph.D. program, Thailand Research Fund, Thailand, and King Mongkut’s University of Technology Thonburi, Thailand.

References

Mohammad Haji-Adolhosseini. 2003. *A Constraint-Based Approach to Information Structure and Prosody Correspondence*. Proc. of The HPSG-2003 Conf., In Stefan Muller (ed.), CSLI Pub., Michigan State Univ., East Lansing, pp. 143-162.

Ewan Klein. 2000. *Prosodic constituency in HPSG*, Grammatical Interfaces in HPSG. In Ronnie Cann, and Philip Miller, ed., CSLI Pub., pp 169-200.

Copestake A. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Pub., Stanford, CA.

Copestake A., Flickinger D., Malouf R., Riehemann S. and Sag I.A. 1995. *Translation using Minimal Recursion Semantics*. Proc. of the The 6th Int’l Conf. on Theoretical and Methodological Issues in Machine Translation (TMI-95), Belgium.

Silverman K., Beckman M. B., Pirelli J., Ostendorf M., Wightman C., Price P., Pierrehumbert J., and Hirschberg J. 1992. *ToBI: A Standard for Labeling English Prosody*. In Proc. of ICSLP’92, Banff, Canada, pages. 867-870.

Steedman M. and Prevost, S. 1994. *Specifying Intonation from Context for Speech Synthesis*. Speech Comm., 15, 1994, 139-153.

Von Heusinger K. 1999. *Intonation and Information Structure*. The Representation of Focus in Phonology and Semantics. Habilitationsschrift, University Konstanz, pp. 125-155.

Paul Boersma and David Weenink 2005. *Praat: doing phonetics by computer*. Inst. of Phonetic Sciences, Univ. of Amsterdam, Netherlands, http://www.praat.org, Oct. 2005.

Ballmer T. and Brennenstuhl W. 1981. *Speech Act Classification*. A study in the Lexical analysis of English speech activity verbs. Springer Series in Language and Comm., Vol.8. Springer Verlag, New York, 1981.

CMU Communicator KAL limited domain. 2002. Language Technologies Inst., Carnegie Mellon Univ., www.festvox.org, Oct 2005.

Sag, Ivan A., Thomas Wasow, and Emily Bender. 2003. *Syntactic Theory: A formal introduction*. CSLI Pub., Univ. of Chicago Press.

Extraction of Tree Adjoining Grammars from a Treebank for Korean

Jungyeul Park
UFR Linguistique
Laboratoire de linguistique formelle
Université Paris VII - Denis Diderot
jungyeul.park@linguist.jussieu.fr

Abstract

We present the implementation of a system which extracts not only lexicalized grammars but also feature-based lexicalized grammars from Korean *Sejong* Treebank. We report on some practical experiments where we extract TAG grammars and tree schemata. Above all, full-scale syntactic tags and well-formed morphological analysis in *Sejong* Treebank allow us to extract syntactic features. In addition, we modify Treebank for extracting lexicalized grammars and convert lexicalized grammars into tree schemata to resolve limited lexical coverage problem of extracted lexicalized grammars.

1 Introduction

An electronic grammar is an interface between the complexity and the diversity of natural language and the regularity and the effectiveness of a language processing, and it is one of the most important elements in the natural language processing. Since traditional manual grammar development is a time-consuming and labor-intensive task, many efforts for automatic and semi-automatic grammar development have been taken during last decades.

Automatic grammar development means that a system extracts a grammar from a Treebank which has an implicit Treebank grammar. The grammar extraction system takes syntactically analyzed sentences as an input and produces a target grammar. The extracted grammar would be same as the Treebank grammar or be different depending on the user's specific purpose. The automatically extracted grammar has the advantage of the coherence of extracted grammars and the rapidity of its development. However, as it always depends on the Treebank which the extraction system uses, its coverage could be limited to the scale of a Treebank. Moreover, the reliable Treebank would be hardly found, especially in public domain.

Semi-automatic grammar development means that a system generates the grammar using the description of the language-specific syntactic (or linguistic) variations and its constraints. A metagrammar in Candito (1999) and a tree description in Xia (2001) are good examples of a semi-automatic grammar development. Even using semi-automatic grammar development, we need the good description of linguistic phenomena for specific language which requires very high level knowledge of linguistics and the semi-automatically generated grammars would easily have an overflow problem.

Since we might extract the grammar automatically without many efforts if a reliable Treebank is provided, in this paper we implement a system which extracts a Lexicalized Tree Adjoining Grammar and a Feature-based Lexicalized Tree Adjoining Grammar from Korean *Sejong* Treebank (SJTree). SJTree contains 32,054 *eojeol*s (the unity of segmentation in the Korean sentence), that is, 2,526 sentences. SJTree uses 43 part-of-speech tags and 55 syntactic tags.

Even though there are many previous works for extracting grammars from a Treebank, extracting syntactic features is tried for the first time. 55 full-scale syntactic tags and well-formed morphological analysis in SJTree allow us to extract syntactic features automatically and to develop FB-LTAG.

First, we briefly present features structures which are focused on FB-LTAG and other previous works for extracting a grammar from a Treebank. Then, we explain our grammar extraction scheme and report experimental results. Finally, we discuss the conclusion.

2 Feature structures and previous works on extracting grammars from a Treebank

A feature structure is a way of representing grammatical information. Formally feature structure consists of a specification of a set of features, each of which is paired with a particular value (Sag *et al.*, 2003). In a unification frame, a feature structure is associated with each node in an elementary tree (Vijay-Shanker and Joshi, 1991). This feature structure contains information about how the node interacts with other nodes in the tree. It consists of a top part, which generally contains information relating to the super-node, and a bottom part, which generally contains information relating to the sub-node (Han *et al.*, 2000).

In FB-LTAG, the feature structure of a new node created by substitution inherits the union of the features of the original nodes. The top feature of new node is the union of the top features ($f_1 \cup f$) of the two original nodes, while the bottom feature of the new node is simply the bottom feature (g_1) of the top node of the substituting tree since the substitution node has no bottom feature as shown in Figure 1.

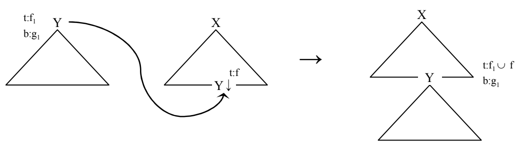


Figure 1. Substitution in FB-LTAG

The node being adjoined into splits and its top feature (f) unifies with the top feature (f_1) of the root adjoining node, while its bottom feature (g) unifies with the bottom feature (g_2) of the foot adjoining node as shown in Figure 2.

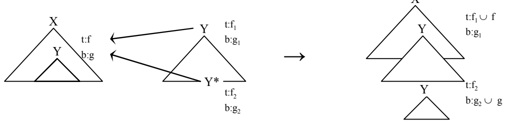


Figure 2. Adjunction in FB-LTAG

Several works for extracting grammars, especially for TAG formalism are proposed. Chen (2001) extracted lexicalized grammars from English Penn Treebank and there are other works based on Chen's procedure such as Johansen (2004) and Nasr (2004) for French and Habash and Rambow (2004) for Arabic. Chiang (2000) used Tree Insertion Grammars, one variation of TAG formalism for his extraction system from English Penn Treebank. Xia *et al.* (2000) developed the uniform method of a grammar extraction for English, Chinese and Korean. Neumann (2003) extracted Lexicalized Tree Grammars from English Penn Treebank for English and from NEGRA Treebank for German. As mentioned above, none of these works tried to extract syntactic features for FB-LTAG.

3 Grammar extraction scheme

Before extracting a grammar automatically, we transform the bracket structure sentence in SJTree into a tree data structure. Afterward, using depth-first algorithm for a tree traverse, we determine a head and the type of operations (substitution or adjunction) for children nodes of the given node if the given node is a non-terminal node.

3.1 Determination of a head

For the determination of a head, we assume the right-most child node as a head among its sibling nodes in end-focus languages like Korean. For instance, the second NP is marked as a head in [NP NP] composition while the first NP is marked for adjunction operation for the extracted grammar G_1 which uses *eojeol*s directly without modification of SJTree (see the section 4 for the detail of extraction experiments). Likewise, in [VP@VV VP@VX] composition where the first VP has a VV (verb) anchor and the last VP has a VX (auxiliary verb) anchor, a principal verb in the first VP could be marked for adjunction operation and an auxiliary verb in the second VP would be a head, that is, the extracted auxiliary verb tree has every argument of whole sentence. This phenomenon could be explained by argument composition. Head nodes of the extracted grammar for a verb *balpyoha.eoss.da* ('announced') in (1) are in bold face in Figure 3 which represents bracketed sentence structure in SJTree

(1) 일본 외무성은 즉각 해명 성명을 발표했다.

ilbon	*oimuseong.eun*
Japan	ministy_of_foreign_affairs.Nom
jeukgak	*haemyeng*
immediately	elucidation
seongmyeng.eul	*balpyo.ha.eoss.da*
declaration.Acc	announce.Pass.Ter

‘The ministry of foreign affairs in Japan immediately announced their elucidation.’

```
(S    (NP_SBJ    (NP ilbon/NNP)
                 (NP_SBJ oimuseong/NNG+eun/JX))
      (VP    (AP jeukgak/MAG)
             (VP    (NP_OBJ    (NP haemyeng/NNG)
                               (NP_OBJ seonmyeng/NNG+eul/JKO))
                    (VP balpyo/NNG+ha/XSV+eoss/EP+da/EF+./SF))))
```

Figure 3. Bracketed sentence in SJTree for (1)

3.2 Distinction between substitution and adjunction operations

Unlike other Treebank corpora such as English Penn Treebank and French Paris 7 Treebank, full-scale syntactic tags in SJTree allow us to easily determine which node would be marked for substitution or adjunction operations. Among 55 syntactic tag in SJTree, nodes labeled with NP (noun phrase), S (sentence), VNP (copular phrase) and VP (verb phrase) which end with _CMP (attribute), _OBJ (object), and _SJB (subject) would be marked for substitution operation, and nodes labeled with the other syntactic tags except a head node would be marked for adjunction operation. In this distinction, some VNP and VP phrases might be marked for substitution operation, which means that VNP and VP phrases are arguments of a head, because SJTree labels VNP and VP instead of NP for the nominalization forms of VNP and VP. In Figure 4, for example, NP_SBJ and NP_OBJ nodes are marked for substitution operation and AP node is marked for adjunction operation.

Children nodes marked for substitution operation are replace by substitution terminal nodes (e.g. NP_SBJ↓) and calls recursively the extraction procedure with its subtree where a root node is the child node itself. Children nodes marked for adjunction operation are removed from the main tree and also calls recursively the extraction procedure with its subtree where we add its parent node of a given child node as a root node and a sibling node as a foot node (e.g. VP*). As defined in the TAG formalism, the foot node has the same label as the root node of the subtree for an adjunction operation.

3.3 Reducing trunk

Extracted grammars as explained above are not always “correct” TAG grammar. Since nodes marked for adjunction operation are removed, there remain intermediate nodes in the main tree. In this case, we remove these redundant nodes. Figure 4 shows how to remove the redundant intermediate nodes from the extracted tree for a verb *balpyoha.eoss.da* (‘announced’) in (1).

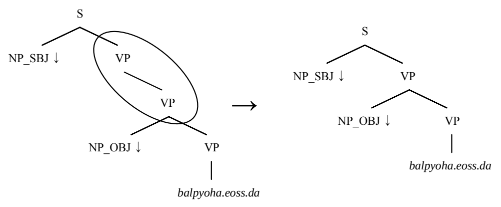


Figure 4. Removing redundant intermediate nodes from extracted trees

3.4 Extracting features

55 full-scale syntactic tags and morphological analysis in SJTree allow us to extract syntactic features automatically and to develop FB-LTAG. Automatically extracted FB-LTAG grammars eventually use reduced tagset because FB-LTAG grammars contain their syntactic information in features structures. For example, NP_SBJ syntactic tag in LTAG is changed into NP and a syntactic feature <case=subject> is added. Therefore, we use actually 13 reduced tagset for FB-LTAG grammars. From full-scale syntactic tags which end with _SBJ (subject), _OBJ (object) and _CMP (attribute), we extract <case> features which describe argument structures in the sentence.

Alongside <case> features, we also extract <mode> and <tense> from morphological analyses in SJTree. Since however morphological analyses for verbal and adjectival endings in SJTree are simply divided into EP, EF and EC which mean non-final endings, final endings and conjunctive endings, respectively, <mode> and <tense> features are not extracted directly from SJTree. In this paper, we analyze 7 non-final endings (EP) and 77 final endings (EF) used in SJTree to extract automatically <mode> and <tense> features. In general, EF carries <mode> inflections, and EP carries <tense> inflections. Conjunctive endings (EC) are not concerned with <mode> and <tense> features and we only extract <ec> features with its string value. <ef> and <ep> features are also extracted with their string values. Some of non-final endings like *si* are extracted as <hor> features which have honorary meaning. In extracted FB-LTAG grammars, we present their lexical heads in a bare infinitive with morphological features such as <ep>, <ef> and <ec> which make correspond with its inflected forms.

<det> is another automatically extractable feature in SJTree and it is extracted from both syntactic tag and morphological analysis unlike other extracted features. For example, while <det=-> is extracted from dependant nouns which always need modifiers (extracted by morphological analyses), <det=+> is extracted from _MOD phrases (extracted by syntactic tags). From syntactic tag DP which contains MMs (determinative or demonstrative), <det=+> is also extracted[1].

The actual procedure of feature extraction is implemented by 2 phases. In the first phase, we convert syntactic tags and morphological analysis into feature structure as explained above. In the second phase, we complete feature structure onto nodes of dorsal spine. For example, we put the same feature of VV bottom onto VV top, VP top/bottom and S bottom because nodes in dorsal spine share certain number of feature of VV bottom. The initial tree for a verb *balpyoha.eoss.da* is completed like Figure 5 for a FB-LTAG (see Park (2006) for details).

[1] Korean does not need features <person> as in English and <gender > or <number> as in French. Han *et al*. (2000) proposed several features for Korean FBLTAG which we do not use in this paper, such as <adv-pp>, <top> and < aux-pp> for nouns and <clause-type> for predicates. While postpositions are separated from *eojeol* during our grammar extraction procedure, Han *el al*. considered them as "one" inflectional morphology of noun phrase *eojeol*. As we will explain the reason why we separate postpositions from *eojeol* in the section 4, the separation of postpositions would be much efficient for the lexical coverage of extracted grammars. In Han *et al*. <adv-pp> simply contains string value of adverbial postpositions. <aux-pp> adds semantic meaning of auxiliary postpositions such as *only*, *also* etc. which we can not extract automatically from SJTree or other Korean Treebank corpora because syntactically annotated Treebank corpora generally do not contain such semantic information. <top> marks the presence or absence of a topic marker in Korean like *neun*, however topic markers are annotated like a subject in SJTree which means that only <case=subject> is extracted for topic markers. <clause-type> indicates the type of the clause which has its values such as main, coord(inative), subordi(native), adnom(inal), nominal, aux-connect. Since the distinction of the type of the clause is very vague except main clause in Korea, we do not adopt this feature. Instead <ef> is extracted if a clause type is a main clause and <ec> is extracted for other type.

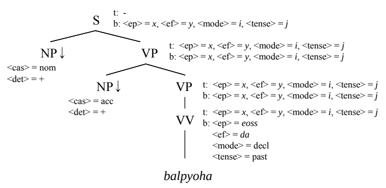

S
t: -
b: <ep> = x, <ef> = y, <mode> = i, <tense> = j
NP↓
<cas> = nom
<det> = +
VP
t: <ep> = x, <ef> = y, <mode> = i, <tense> = j
b: <ep> = x, <ef> = y, <mode> = i, <tense> = j
NP↓
<cas> = acc
<det> = +
VP
t: <ep> = x, <ef> = y, <mode> = i, <tense> = j
b: <ep> = x, <ef> = y, <mode> = i, <tense> = j
VV
t: <ep> = x, <ef> = y, <mode> = i, <tense> = j
b: <ep> = eoss
<ef> = da
<mode> = decl
<tense> = past
balpyoha

Figure 5. Extracted FB-LTAG grammar for *balpyoha.eoss.da* ('announced')

4 Extraction experiments and results

4.1 Extraction of lexicalized trees

In this paper, we extract not only lexicalized trees without modification of a Treebank, but also extract grammars with modifications of a Treebank using some constraints to improve the lexical coverage in extracted grammars.

- G_1: Using *eojeol*s directly without modification of SJTree.
- G_2: Separating symbols and postpositions from *eojeol*s. Separated symbols are extracted and divided into α and β trees based on their types. Every separated postposition is α tree. Complex postpositions consisted of two or more postpositions are extracted like one α tree[2]. Finally, converting NP β trees into α trees and removing syntactic tag in NP α trees.

Figure 6 and 7 show extracted lexicalized grammars G_1 and G_2 from (1) respectively. Theoretically extracting order is followed by word order in the sentence.

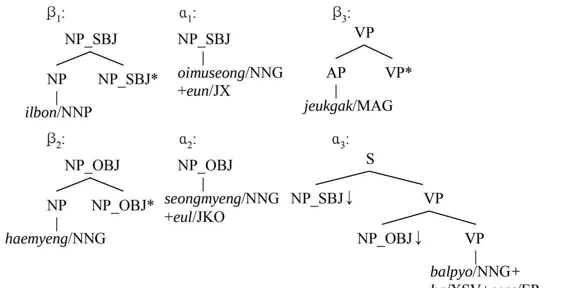
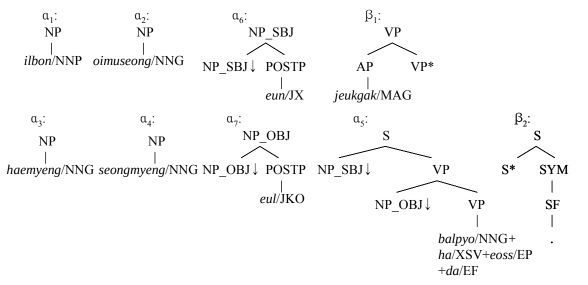


Figure 6. Extracted lexicalized grammars G_1

[2] For extracting trees of symbols and of postposition, we newly add SYM and POSTP syntactic tags which SJTree does not use. See Figure 11 for extracted symbol and postposition trees.

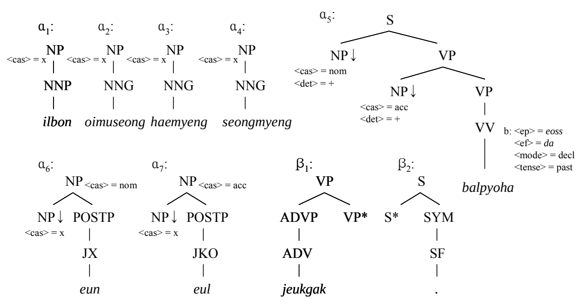

α1: NP ilbon/NNP
α2: NP oimuseong/NNG
α6: NP_SBJ NP_SBJ↓ POSTP eun/JX
β1: VP AP VP* jeukgak/MAG
α3: NP haemyeng/NNG
α4: NP seongmyeng/NNG
α7: NP_OBJ NP_OBJ↓ POSTP eul/JKO
α5: S NP_SBJ↓ VP NP_OBJ↓ VP balpyo/NNG+ ha/XSV+eoss/EP +da/EF
β2: S S* SYM SF .
Figure 7. Extracted lexicalized grammars G_2

4.2 Extraction of feature-based lexicalized trees

We extract feature-based lexicalized trees using reduced tagset because FB-LTAG grammars contain their syntactic information in features structures. Extracted grammars G_3 remove syntactic tags, eventually use reduced tagset, add extracted feature structures and use infinitive forms as lexical anchor.

- G_3: Using reduced tagset and a lexical anchor is an infinitive and adding extracted feature structures.

G_3 row in Table 1 below shows the results of extraction procedures above. Figure 8 shows extracted feature-based lexicalized grammars G_3 from (1)


α1: NP <cas> = x NNP ilbon
α2: NP <cas> = x NNG oimuseong
α3: NP <cas> = x NNG haemyeng
α4: NP <cas> = x NNG seongmyeng
α5: S NP↓ <cas> = nom <det> = + VP NP↓ <cas> = acc <det> = + VP VV balpyoha
b: <ep> = eoss <ef> = da <mode> = decl <tense> = past
α6: NP <cas> = nom NP↓ <cas> = x POSTP JX eun
α7: NP <cas> = acc NP↓ <cas> = x POSTP JKO eul
β1: VP ADVP VP* ADV jeukgak
β2: S S* SYM SF .
Figure 8. Extracted feature-based lexicalized grammars G_3 [3].

	# of `ltrees` (lexicalized tree)	Average frequencies per `ltrees`
G_1	18,080	1.38
G_2	15,551	2.57
G_3	12,429	3.21

Table 1. Results of experiments in extracting lexicalized and feature-based lexicalized grammars

[3] To simplify the figure, we note only feature structure which is necessary to understand.

4.3 Extraction of tree schemata

As mentioned in the Introduction, one of the most serious problems in automatic grammar extraction is its limited lexical coverage. To resolve this problem, we enlarge our extracted lexicalized grammars using templates which we call tree schemata. The lexical anchor is removed from extracted grammars and anchor mark is replaced to form tree schemata (for example, @NNG where the lexicalized anchor in extracted lexicalized grammars is a common noun). The number of tree schemata is much reduced against that of lexicalized grammars. Table 2 shows the number of template trees and the average frequency for each template grammars. T_1 means G_1's tree schemata.

	# of tree schemata	Average frequencies per tree schemata
T_1	1,158	21.55
T_2	1,077	37.05
T_3	385	103.65

Table 2. Results of experiments in converting template grammars

5 Evaluations

First of all, the lexical coverage for G_1 and G_2 is tested on the part of *Sejong* corpus which contains about 770,000 "morphologically analyzed" *eojeol*s. After modification of SJTree, the extracted grammar G_2 is increased to 17.8 % compared with G_1 for its lexical coverage. G_2 and G_3 have same lexical coverage since they have same lexical entries.

Extracted grammars in this paper are evaluated by its size and its coverage. The size of grammars means tree schemata according to the number of sentences as shown in Figure 9. The coverage of grammar is the number of occurrences of unknown tree schemata in the corpus by the total occurrences of tree schemata as shown in Table 3.

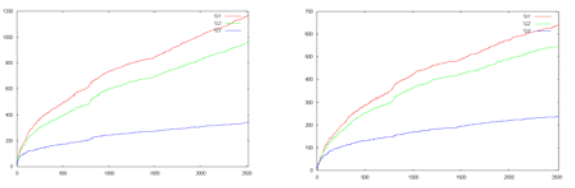

(a) Threshold =1 (b) Threshold =2

Figure 9. The size of grammars

	Threshold = 1	Threshold = 2
G_1	0.9326	0.9591
G_2	0.9326	0.9525
G_3	0.9579	0.9638

Table 3. Coverage of grammars: 90% of training set (2,273 sentences) and 10% of test set (253 sentences)

We manually overlap our 163 tree schemata for predicates from T_3, which contain 14 subcategorization frames with 11 subcategorization frames of a FB-LTAG grammar proposed in Han *et al.* (2000) to evaluate the coverage of hand-crafted grammars[4]. Our extracted template grammars cover 72.7 % of their hand-crafted subcategorization frames[5].

6 Conclusion

In this paper, we have presented a system for automatic grammar extraction that produces lexicalized and feature-based lexicalized grammars from a Treebank. Also, to resolve the problem of limited lexical coverage of extracted grammars, we separated symbols and postposition, and then converted these grammars into template grammars. Extracted grammars and lexical-anchor-less template grammars might be used for parsers to analyze the Korean sentences and frequency information might be used to remove ambiguities among possible syntactic analyses of parsers.

References

Candito, Marie-Hélène. 1999. *Organisation modulaire et paramétrable de grammaire électronique lexicalisées*. Ph.D. thesis, Université Paris 7.

Chen, John. 2001. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

Chiang, David. 2000. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Data Oriented Parsing*, CSLI Publication, pp. 299-316.

Habash, Nizar and Owen Rambow. 2004. Extracting a Tree Adjoining Grammar from the Penn Arabic Treebank. In *Proceedings of Traitement Automatique du Langues Naturelles (TALN-04)*. Fez, Morocco, 2004.

Han, Chunghye, Juntae Yoon, Nari Kim, and Martha Palmer. 2000. *A Feature-Based Lexicalized Tree Adjoining Grammar for Korean*. IRCS Technical Report 00-04. University of Pennsylvania.

Johansen, Ane Dybro. 2004. *Extraction des grammaires LTAG à partir d'un corpus étiquette syntaxiquement*. DEA mémoire, Université Paris 7.

Nasr, Alexis. 2004. *Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement*. Habilitation à diriger des recherches, Université Paris 7.

Neumann, Günter. 2003. A Uniform Method for Automatically Extracting Stochastic Lexicalized Tree Grammar from Treebank and HPSG, In A. Abeillé (ed) *Treebanks: Building and Using Parsed Corpora*, Kluwer, Dordrecht.

Park, Jungyeul. 2006. *Extraction d'une grammaire d'arbres adjoints à partir d'un corpus arboré pour le coréen*. Ph.D. thesis, Université Paris 7.

Sag, Ivan A., Thomas Wasow, and Emily M. Bender. 2003. *Syntactic Theory: A Formal Introduction*, 2nd ed. CSLI Lecture Notes.

Vijay-Shanker, K. and Aravind K. Joshi. 1991. Unification Based Tree Adjoining Grammar, in J. Wedekind ed., *Unification-based Grammars*, MIT Press, Cambridge, Massachusetts.

Xia, Fei, Martha Palmer, and Aravind K. Joshi. 2000. A Uniform Method of Grammar Extraction and Its Application. In *The Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong, Oct 7-8, 2000.

Xia, Fei. 2001. *Automatic Grammar Generation from Two Different Perspectives*. Ph.D. thesis, University of Pennsylvania, PA.

[4] Our extracted tree schemata contain not only subcategorization frames but also some phenomena of syntactic variations, the number of lexicalized trees and the frequency information while Han *el al*. (2000) only presents subcategorization frames and some phenomena.

[5] Three subcategorization frames in Han *el al*. (2000) which contain prepositional phrases are not covered by our extracted tree schemata. Generally, prepositional phrases in SJTree are labeled with _AJT which is marked for adjunction operation. Since there is no difference between noun adverbial phrase and prepositional phrases in SJTree like [S *na.neun* [NP_AJT *ojeon.e* 'morning'] [NP_AJT *hakgyo.e* 'to school'] *ga.ss.da*] ('I went to school this morning'), we do not consider _AJT phrases as arguments.

Parsing and Subcategorization Data

Jianguo Li
Department of Linguistics
The Ohio State University
Columbus, OH, USA
`jianguo@ling.ohio-state.edu`

Abstract

In this paper, we compare the performance of a state-of-the-art statistical parser (Bikel, 2004) in parsing written and spoken language and in generating subcategorization cues from written and spoken language. Although Bikel's parser achieves a higher accuracy for parsing written language, it achieves a higher accuracy when extracting subcategorization cues from spoken language. Additionally, we explore the utility of punctuation in helping parsing and extraction of subcategorization cues. Our experiments show that punctuation is of little help in parsing spoken language and extracting subcategorization cues from spoken language. This indicates that there is no need to add punctuation in transcribing spoken corpora simply in order to help parsers.

1 Introduction

Robust statistical syntactic parsers, made possible by new statistical techniques (Collins, 1999; Charniak, 2000; Bikel, 2004) and by the availability of large, hand-annotated training corpora such as WSJ (Marcus et al., 1993) and Switchboard (Godefrey et al., 1992), have had a major impact on the field of natural language processing. There are many ways to make use of parsers' output. One particular form of data that can be extracted from parses is information about subcategorization. Subcategorization data comes in two forms: subcategorization frame (SCF) and subcategorization cue (SCC). SCFs differ from SCCs in that SCFs contain only arguments while SCCs contain both arguments and adjuncts. Both SCFs and SCCs have been crucial to NLP tasks. For example, SCFs have been used for verb disambiguation and classification (Schulte im Walde, 2000; Merlo and Stevenson, 2001; Lapata and Brew, 2004; Merlo et al., 2005) and SCCs for semantic role labeling (Xue and Palmer, 2004; Punyakanok et al., 2005).

Current technology for automatically acquiring subcategorization data from corpora usually relies on statistical parsers to generate SCCs. While great efforts have been made in parsing written texts and extracting subcategorization data from written texts, spoken corpora have received little attention. This is understandable given that spoken language poses several challenges that are absent in written texts, including disfluency, uncertainty about utterance segmentation and lack of punctuation. Roland and Jurafsky (1998) have suggested that there are substantial subcategorization differences between written corpora and spoken corpora. For example, while written corpora show a much higher percentage of passive structures, spoken corpora usually have a higher percentage of zero-anaphora constructions. We believe that subcategorization data derived from spoken language, if of acceptable quality, would be of more value to NLP tasks involving a syntactic analysis of spoken language, but we do not pursue it here.

The goals of this study are as follows:

1. Test the performance of Bikel's parser in parsing written and spoken language.

2. Compare the accuracy level of SCCs generated from parsed written and spoken language. We hope that such a comparison will shed some light on the feasibility of acquiring SCFs from spoken language using the cur-

rent SCF acquisition technology initially designed for written language.

3. Explore the utility of punctuation[1] in parsing and extraction of SCCs. It is generally recognized that punctuation helps in parsing written texts. For example, Roark (2001) finds that removing punctuation from both training and test data (WSJ) decreases his parser's accuracy from 86.4%/86.8% (LR/LP) to 83.4%/84.1%. However, spoken language does not come with punctuation. Even when punctuation is added in the process of transcription, its utility in helping parsing is slight. Both Roark (2001) and Engel et al. (2002) report that removing punctuation from both training and test data (Switchboard) results in only 1% decrease in their parser's accuracy.

2 Experiment Design

Three models will be investigated for parsing and extracting SCCs from the parser's output:

1. **punc**: leaving punctuation in both training and test data.

2. **no-punc**: removing punctuation from both training and test data.

3. **punc-no-punc**: removing punctuation from only test data.

Following the convention in the parsing community, for written language, we selected sections 02-21 of WSJ as training data and section 23 as test data (Collins, 1999). For spoken language, we designated section 2 and 3 of Switchboard as training data and files of sw4004 to sw4135 of section 4 as test data (Roark, 2001). Since we are also interested in extracting SCCs from the parser's output, we eliminated from the two test corpora all sentences that do not contain verbs. Our experiments proceed in the following three steps:

1. Tag test data using the POS-tagger described in Ratnaparkhi (1996).

2. Parse the POS-tagged data using Bikel's parser.

[1]We use punctuation to refer to sentence-internal punctuation unless otherwise specified.

label	clause type	desired SCCs
S	gerundive	(NP)-GERUND
	small clause	NP-NP, (NP)-ADJP
	control	(NP)-INF-*to*
SBAR	control	(NP)-INF-*wh-to*
	with a complementizer	(NP)-S-*wh*, (NP)-S-*that*
	without a complementizer	(NP)-S-*that*

Table 1: SCCs for different clauses

3. Extract SCCs from the parser's output. The extractor we built first locates each verb in the parser's output and then identifies the syntactic categories of all its sisters and combines them into an SCC. However, there are cases where the extractor has more work to do.

   - Finite and Infinite Clauses: In the Penn Treebank, **S** and **SBAR** are used to label different types of clauses, obscuring too much detail about the internal structure of each clause. Our extractor is designed to identify the internal structure of different types of clause, as shown in Table 1.
   - Passive Structures: As noted above, Roland and Jurafsky (Roland and Jurafsky, 1998) have noticed that written language tends to have a much higher percentage of passive structures than spoken language. Our extractor is also designed to identify passive structures from the parser's output.

3 Experiment Results

3.1 Parsing and SCCs

We used EVALB measures Labeled Recall (LR) and Labeled Precision (LP) to compare the parsing performance of different models. To compare the accuracy of SCCs proposed from the parser's output, we calculated SCC Recall (SR) and SCC Precision (SP). SR and SP are defined as follows:

$$SR = \frac{\text{number of correct cues from the parser's output}}{\text{number of cues from treebank parse}} \quad (1)$$

$$SP = \frac{\text{number of correct cues from the parser's output}}{\text{number of cues from the parser's output}} \quad (2)$$

$$\text{SCC Balanced F-measure} = \frac{2 * SR * SP}{SR + SP} \quad (3)$$

The results for parsing WSJ and Switchboard and extracting SCCs are summarized in Table 2.

The LR/LP figures show the following trends:

WSJ		
model	LR/LP	SR/SP
punc	87.92%/88.29%	76.93%/77.70%
no-punc	86.25%/86.91%	76.96%/76.47%
punc-no-punc	82.31%/83.70%	74.62%/74.88%
Switchboard		
model	LR/LP	SR/SP
punc	83.14%/83.80%	79.04%/78.62%
no-punc	82.42%/83.74%	78.81%/78.37%
punc-no-punc	78.62%/80.68%	75.51%/75.02%

Table 2: Results of parsing and extraction of SCCs

1. Roark (2001) showed LR/LP of 86.4%/86.8% for punctuated written language, 83.4%/84.1% for unpunctuated written language. We achieve a higher accuracy in both punctuated and unpunctuated written language, and the decrease if punctuation is removed is less

2. For spoken language, Roark (2001) showed LR/LP of 85.2%/85.6% for punctuated spoken language, 84.0%/84.6% for unpunctuated spoken language. We achieve a lower accuracy in both punctuated and unpunctuated spoken language, and the decrease if punctuation is removed is less. The trends in (1) and (2) may be due to parser differences, or to the removal of sentences lacking verbs.

3. Unsurprisingly, if the test data is unpunctuated, but the models have been trained on punctuated language, performance decreases sharply.

In terms of the accuracy of extraction of SCCs, the results follow a similar pattern. However, the utility of punctuation turns out to be even smaller. Removing punctuation from both training and test data results in a less than 0.3% drop in the accuracy of SCC extraction.

Figure 1 exhibits the relation between the accuracy of parsing and that of extracting SCCs. If we consider WSJ and Switchboard individually, there seems to exist a positive correlation between the accuracy of parsing and that of extracting SCCs. In other words, higher LR/LP indicates higher SR/SP. However, Figure 1 also shows that although the parser achieves a higher F-measure value for paring WSJ, it achieves a higher F-measure value when generating SCCs from Switchboard.

The fact that the parser achieves a higher accuracy for extracting SCCs from Switchboard than WSJ merits further discussion. Intuitively, it seems to be true that the shorter an SCC is, the more likely that the parser is to get it right. This intuition is confirmed by the data shown in Figure 2. Figure 2 plots the accuracy level of extracting SCCs by SCC's length. It is clear from Figure 2 that as SCCs get longer, the F-measure value drops progressively for both WSJ and Switchboard. Again, Roland and Jurafsky (1998) have suggested that one major subcategorization difference between written and spoken corpora is that spoken corpora have a much higher percentage of the zero-anaphora construction. We then examined the distribution of SCCs of different length in WSJ and Switchboard. Figure 3 shows that SCCs of length 0[2] account for a much higher percentage in Switchboard than WSJ, but it is always the other way around for SCCs of non-zero length. This observation led us to believe that the better performance that Bikel's parser achieves in extracting SCCs from Switchboard may be attributed to the following two factors:

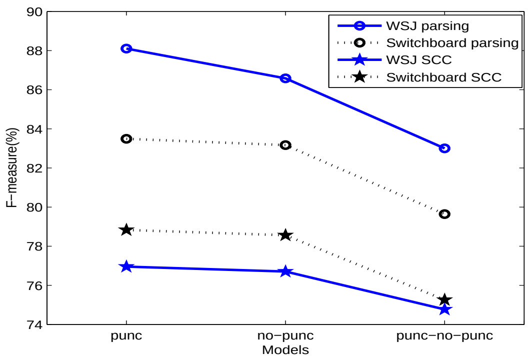


Figure 1: F-measure for parsing and extraction of SCCs

1. Switchboard has a much higher percentage of SCCs of length 0.

2. The parser is very accurate in extracting shorter SCCs.

3.2 Extraction of Dependents

In order to estimate the effects of SCCs of length 0, we examined the parser's performance in retrieving dependents of verbs. Every constituent (whether an argument or adjunct) in an SCC generated by the parser is considered a dependent of

[2]Verbs have a length-0 SCC if they are intransitive and have no modifiers.

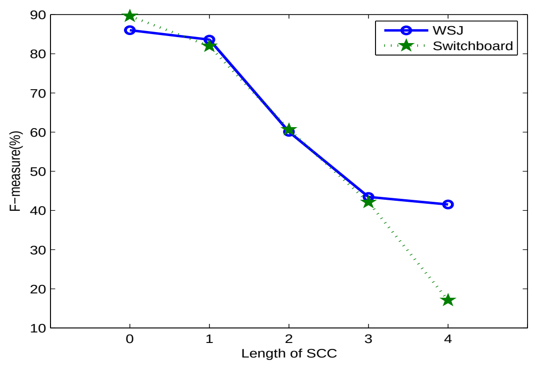


Figure 2: F-measure for SCCs of different length

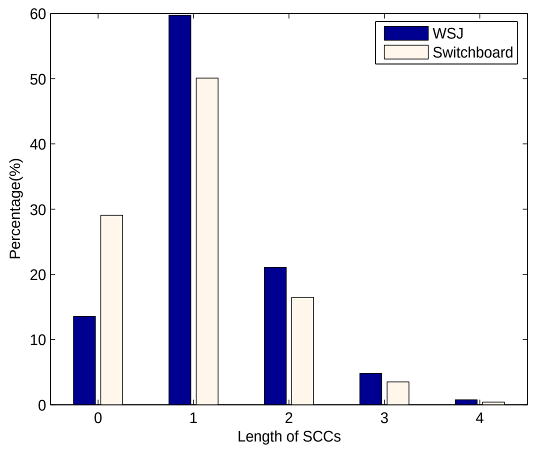


Figure 3: Distribution of SCCs by length

that verb. SCCs of length 0 will be discounted because verbs that do not take any arguments or adjuncts have no dependents[3]. In addition, this way of evaluating the extraction of SCCs also matches the practice in some NLP tasks such as semantic role labeling (Xue and Palmer, 2004). For the task of semantic role labeling, the total number of dependents correctly retrieved from the parser's output affects the accuracy level of the task.

To do this, we calculated the number of dependents shared by between each SCC proposed from the parser's output and its corresponding SCC proposed from Penn Treebank. We based our calculation on a modified version of Minimum Edit Distance Algorithm. Our algorithm works by creating a shared-dependents matrix with one column for each constituent in the target sequence (SCCs proposed from Penn Treebank) and one

[3]We are aware that subjects are typically also considered dependents, but we did not include subjects in our experiments

```
shared-dependents[i,j] = MAX(
shared-dependents[i-1,j],
shared-dependents[i-1,j-1]+1 if target[i] = source[j],
shared-dependents[i-1,j-1] if target[i] != source[j],
shared-dependents[i,j-1])
```

Table 3: The algorithm for computing shared dependents

INF	#5	1	1	2	**3**
ADVP	#4	1	1	2	2
PP-*in*	#3	1	1	2	2
NP	#2	1	1	1	1
NP	#1	1	1	1	1
	#0	#1	#2	#3	#4
		NP	S-*that*	PP-*in*	INF

Table 4: An example of computing the number of shared dependents

row for each constituent in the source sequence (SCCs proposed from the parser's output). Each cell shared-dependent[*i*,*j*] contains the number of constituents shared between the first *i* constituents of the target sequence and the first *j* constituents of the source sequence. Each cell can then be computed as a simple function of the three possible paths through the matrix that arrive there. The algorithm is illustrated in Table 3.

Table 4 shows an example of how the algorithm works with NP-S-*that*-PP-*in*-INF as the target sequence and NP-NP-PP-*in*-ADVP-INF as the source sequence. The algorithm returns 3 as the number of dependents shared by two SCCs.

We compared the performance of Bikel's parser in retrieving dependents from written and spoken language over all three models using Dependency Recall (DR) and Dependency Precision (DP). These metrics are defined as follows:

$$DR = \frac{\text{number of correct dependents from parser's output}}{\text{number of dependents from treebank parse}} \tag{4}$$

$$DP = \frac{\text{number of correct dependents from parser's output}}{\text{number of dependents from parser's output}} \tag{5}$$

$$\text{Dependency F-measure} = \frac{2 * DR * DP}{DR + DP} \tag{6}$$

The results of Bikel's parser in retrieving dependents are summarized in Figure 4. Overall, the parser achieves a better performance for WSJ over all three models, just the opposite of what have been observed for SCC extraction. Interestingly, removing punctuation from both the training and test data actually slightly improves the F-measure.

This holds true for both WSJ and Switchboard. This Dependency F-measure differs in detail from similar measures in (Xue and Palmer, 2004). For present purposes all that matters is the relative value for WSJ and Switchboard.

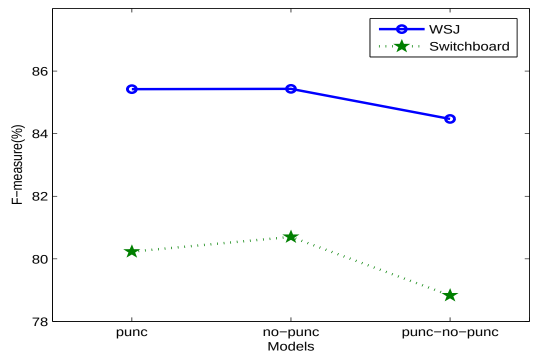


Figure 4: F-measure for extracting dependents

4 Conclusions and Future Work

4.1 Use of Parser's Output

In this paper, we have shown that it is not necessarily true that statistical parsers always perform worse when dealing with spoken language. The conventional accuracy metrics for parsing (LR/LP) should not be taken as the only metrics in determining the feasibility of applying statistical parsers to spoken language. It is necessary to consider what information we want to extract out of parsers' output and make use of.

1. Extraction of SCFs from Corpora: This task usually proceeds in two stages: (i) Use statistical parsers to generate SCCs. (ii) Apply some statistical tests such as the Binomial Hypothesis Test (Brent, 1993) and log-likelihood ratio score (Dunning, 1993) to SCCs to filter out false SCCs on the basis of their reliability and likelihood. Our experiments show that the SCCs generated for spoken language are as accurate as those generated for written language, which suggests that it is feasible to apply the current technology for automatically extracting SCFs from corpora to spoken language.

2. Semantic Role Labeling: This task usually operates on parsers' output and the number of dependents of each verb that are correctly retrieved by the parser clearly affects the accuracy of the task. Our experiments show that the parser achieves a much lower accuracy in retrieving dependents from the spoken language than written language. This seems to suggest that a lower accuracy is likely to be achieved for a semantic role labeling task performed on spoken language. We are not aware that this has yet been tried.

4.2 Punctuation and Speech Transcription Practice

Both our experiments and Roark's experiments show that parsing accuracy measured by LR/LP experiences a sharper decrease for WSJ than Switchboard after we removed punctuation from training and test data. In spoken language, commas are largely used to delimit disfluency elements. As noted in Engel et al. (2002), statistical parsers usually condition the probability of a constituent on the types of its neighboring constituents. The way that commas are used in speech transcription seems to have the effect of increasing the range of neighboring constituents, thus fragmenting the data and making it less reliable. On the other hand, in written texts, commas serve as more reliable cues for parsers to identify phrasal and clausal boundaries.

In addition, our experiment demonstrates that punctuation does not help much with extraction of SCCs from spoken language. Removing punctuation from both the training and test data results in a less than 0.3% decrease in SR/SP. Furthermore, removing punctuation from both the training and test data actually slightly improves the performance of Bikel's parser in retrieving dependents from spoken language. All these results seem to suggest that adding punctuation in speech transcription is of little help to statistical parsers including at least three state-of-the-art statistical parsers (Collins, 1999; Charniak, 2000; Bikel, 2004). As a result, there may be other good reasons why someone who wants to build a Switchboard-like corpus should choose to provide punctuation, but there is no need to do so simply in order to help parsers.

However, segmenting utterances into individual units is necessary because statistical parsers require sentence boundaries to be clearly delimited. Current statistical parsers are unable to handle an input string consisting of two sentences. For example, when presented with an input string as in (1) and (2), if the two sentences are separated by a period (1), Bikel's parser wrongly treats the second sentence as a sentential complement of the main verb *like* in the first sentence. As a result, the extractor generates an SCC NP-S for *like*, which is incorrect. The parser returns the same parse after we removed the period (2) and let the parser parse it again.

(1) I like the long hair. It was back in high school.

(2) I like the long hair It was back in high school.

Hence, while adding punctuation in transcribing a Switchboard-like corpus is not of much help to statistical parsers, segmenting utterances into individual units is crucial for statistical parsers. In future work, we plan to develop a system capable of automatically segmenting speech utterances into individual units.

5 Acknowledgments

This study was supported by NSF grant 0347799. Our thanks go to Chris Brew, Eric Fosler-Lussier, Mike White and three anonymous reviewers for their valuable comments.

References

D. Bikel. 2004. Intricacies of Collin's parsing models. *Computational Linguistics*, 30(2):479–511.

M. Brent. 1993. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(3):243–262.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 2000 Conference of the North American Chapter of the Association for Computation Linguistics*, pages 132–139.

M. Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

D. Engel, E. Charniak, and M. Johnson. 2002. Parsing and disfluency placement. In *Proceedings of 2002 Conference on Empirical Methods of Natural Language Processing*, pages 49–54.

J. Godefrey, E. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP-92*, pages 517–520.

M. Lapata and C. Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–73.

M. Marcus, G. Kim, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

P. Merlo and S. Stevenson. 2001. Automatic verb classification based on statistical distribution of argument structure. *Computational Linguistics*, 27(3):373–408.

P. Merlo, E. Joanis, and J. Henderson. 2005. Unsupervised verb class disambiguation based on diathesis alternations. manuscripts.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the 2nd Midwest Computational Linguistics Colloquium*, pages 15–22.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods of Natural Language Processing*, pages 133–142.

B. Roark. 2001. *Robust Probabilistic Predictive Processing: Motivation, Models, and Applications*. Ph.D. thesis, Brown University.

D. Roland and D. Jurafsky. 1998. How verb subcategorization frequency is affected by the corpus choice. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 1122–1128.

S. Schulte im Walde. 2000. Clustering verbs semantically according to alternation behavior. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 747–753.

N. Xue and M. Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94.

Clavius: Bi-Directional Parsing for Generic Multimodal Interaction

Frank Rudzicz
Centre for Intelligent Machines
McGill University
Montréal, Canada
frudzi@cim.mcgill.ca

Abstract

We introduce a new multi-threaded parsing algorithm on unification grammars designed specifically for multimodal interaction and noisy environments. By lifting some traditional constraints, namely those related to the ordering of constituents, we overcome several difficulties of other systems in this domain. We also present several criteria used in this model to constrain the search process using dynamically loadable scoring functions. Some early analyses of our implementation are discussed.

1 Introduction

Since the seminal work of Bolt (Bolt, 1980), the methods applied to multimodal interaction (MMI) have diverged towards unreconcilable approaches retrofitted to models not specifically amenable to the problem. For example, the representational differences between neural networks, decision trees, and finite-state machines (Johnston and Bangalore, 2000) have limited the adoption of the results using these models, and the typical reliance on the use of whole unimodal sentences defeats one of the main advantages of MMI - the ability to constrain the search using cross-modal information as early as possible.

CLAVIUS is the result of an effort to combine sensing technologies for several modality types, speech and video-tracked gestures chief among them, within the immersive virtual environment (Boussemart, 2004) shown in Figure 1. Its purpose is to comprehend multimodal phrases such as "*put this* $\searrow$ *here* $\searrow$.", for pointing gestures $\searrow$, in either command-based or dialogue interaction.

CLAVIUS provides a flexible, and trainable new bi-directional parsing algorithm on multi-dimensional input spaces, and produces modality-independent semantic interpretation with a low computational cost.

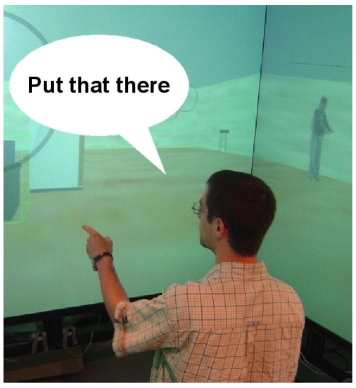


Figure 1: The target immersive environment.

1.1 Graphical Models and Unification

Unification grammars on typed directed acyclic graphs have been explored previously in MMI, but typically extend existing mechanisms not designed for multi-dimensional input. For example, both (Holzapfel et al., 2004) and (Johnston, 1998) essentially adapt Earley's chart parser by representing edges as sets of references to terminal input elements - unifying these as new edges are added to the agenda. In practice this has led to systems that analyze every possible subset of the input resulting in a combinatorial explosion that balloons further when considering the complexities of cross-sentential phenomena such as anaphora, and the effects of noise and uncertainty on speech and gesture tracking. We will later show the extent to which CLAVIUS reduces the size of the search space.

Directed graphs conveniently represent both syntactic and semantic structure, and all partial parses in CLAVIUS , including terminal-level input, are represented graphically. Few restrictions apply, except that arcs labelled CAT and TIME must exist to represent the grammar category and time spanned by the parse, respectively[1]. Similarly, all grammar rules, Γ_i : $LHS \longrightarrow RHS_1\ RHS_2\ ...\ RHS_r$, are graphical structures, as exemplified in Figure 2.

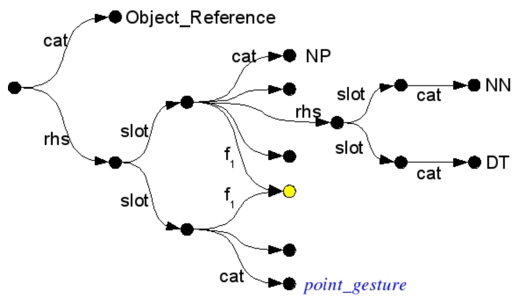


Figure 2: Γ_1 : OBJECT_REFERENCE $\longrightarrow$ NP *click* $\{where(\text{NP} :: f_1) = (click :: f_1)\}$, with NP expanded by Γ_2 : NP $\longrightarrow$ DT NN.

1.2 Multimodal Bi-Directional Parsing

Our parsing strategy combines *bottom-up* and *top-down* approaches, but differs from other approaches to bi-directional chart parsing (Rocio, 1998) in several key respects, discussed below.

1.2.1 Asynchronous Collaborating Threads

A defining characteristic of our approach is that edges are selected *asynchronously* by two concurrent processing threads, rather than serially in a two-stage process. In this way, we can distribute processing across multiple machines, or dynamically alter the priorities given to each thread. Generally, this allows for a more dynamic process where no thread can dominate the other. In typical bi-directional chart parsing the *top-down* component is only activated when the *bottom-up* component has no more legal expansions (Ageno, 2000).

1.2.2 Unordered Constituents

Alhough evidence suggests that deictic gestures overlap *or follow* corresponding spoken pronomials 85-93% of the time (Kettebekov et al, 2002), we must allow for all possible permutations of multi-dimensional input - as in "*put* $\searrow$ *this* $\searrow$ *here.*" vs. "*put this* $\searrow$ *here* $\searrow$.", for example.

We therefore take the unconvential approach of placing no mandatory ordering constraints on constituents, hence the rule Γ_{abc} : $A \longrightarrow$ B C parses the input " C B". We show how we can easily maintain regular temporal ordering in §3.5.

1.2.3 Partial Qualification

Whereas existing bi-directional chart parsers maintain fully-qualified edges by incrementally adding adjacent input words to the agenda, CLAVIUS has the ability to construct parses that instantiate only a subset of their constituents, so Γ_{abc} also parses the input "B", for example. Repercussions are discussed in §3.4 and §4.

2 The Algorithm

CLAVIUS expands parses according to a best-first process where newly expanded edges are ordered according to trainable criteria of multimodal language, as discussed in §3. Figure 3 shows a component breakdown of CLAVIUS 's software architecture. The sections that follow explain the flow of information through this system from sensory input to semantic interpretation.

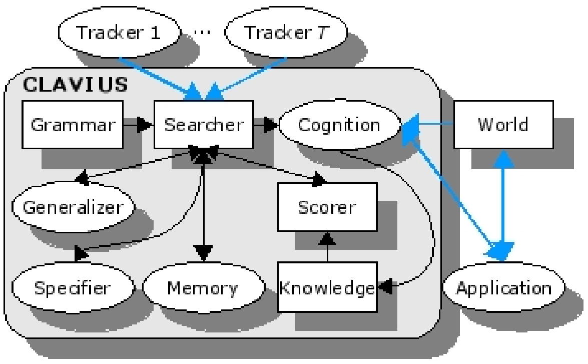


Figure 3: Simplified information flow between fundamental software components.

2.1 Lexica and Preprocessing

Each unique input modality is asynchronously monitored by one of T TRACKERS, each sending an n-best list of lexical hypotheses to CLAVIUS for any activity as soon as it is detected. For example, a gesture tracker (see Figure 4a) parametrizes the gestures *preparation*, *stroke/point*, and *retraction* (McNeill, 1992), with values reflecting spatial positions and velocities of arm motion, whereas

[1]Usually this timespan corresponds to the real-time occurrence of a speech or gestural event, but the actual semantics are left to the application designer

our speech tracker parametrises words with part-of-speech tags, and prior probabilities (see Figure 4b). Although preprocessing is reduced to the identification of lexical tokens, this is more involved than simple lexicon lookup due to the modelling of complex signals.

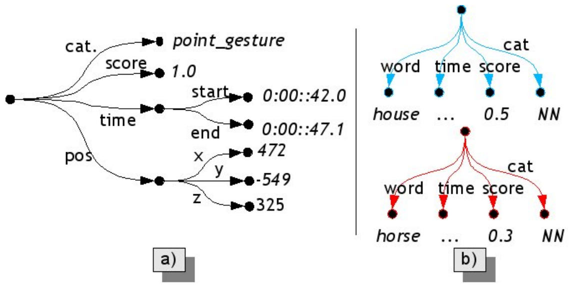


Figure 4: Gestural (a) and spoken (b) 'words'.

2.2 Data Structures

All TRACKERS write their hypotheses directly to the first of three SUBSPACES that partition all partial parses in the search space. The first is the GENERALISER's subspace, $\Xi^{[G]}$, which is monitored by the GENERALISER thread - the first part of the parser. All new parses are first written to $\Xi^{[G]}$ before being moved to the SPECIFIER's active and inactive subspaces, $\Xi^{[SAct]}$, and $\Xi^{[SInact]}$, respectively. Subspaces are optimised for common operations by organising parses by their scores and grammatical categories into depth-balanced search trees having the heap property. The best partial parse in each subspace can therefore be found in $O(1)$ amortised time.

2.3 Generalisation

The GENERALISER monitors the best partial parse, Ψ_g, in $\Xi^{[G]}$, and creates new parses Ψ_i for all grammar rules Γ_i having CATEGORY(Ψ_g) on the right-hand side. Effectively, these new parses are instantiations of the relevant Γ_i, with one constituent unified to Ψ_g. This provides the impetus towards sentence-level parses, as simplified in Algorithm 1 and exemplified in Figure 5. Naturally, if rule Γ_i has more than one constituent ($c > 1$) of type CATEGORY(Ψ_g), then c new parses are created, each with one of these being instantiated.

Since the GENERALISER is activated as soon as input is added to $\Xi^{[G]}$, the process is *interactive* (Tomita, 1985), and therefore incorporates the associated benefits of efficiency. This is contrasted with the all-paths bottom-up strategy in GEMINI (Dowding et al, 1993) that finds all admissable edges of the grammar.

Algorithm 1: Simplified Generalisation

```
Data: Subspace $\Xi^{[G]}$, grammar $\Gamma$
while data remains in $\Xi^{[G]}$ do
    $\Psi_g$ := highest scoring graph in $\Xi^{[G]}$
    foreach rule $\Gamma_i$ s.t. Cat $(\Psi_g) \in RHS(\Gamma_i)$
    do
        $\Psi_i$ := Unify $(\Gamma_i, [\bullet \rightarrow_{\text{RHS}} \bullet \Rightarrow \Psi_g])$
        if $\exists \Psi_i$ then
            Apply Score $(\Psi_i)$ to $\Psi_i$
            Insert $\Psi_i$ into $\Xi^{[G]}$
    Move $\Psi_g$ into $\Xi^{[SAct]}$
```

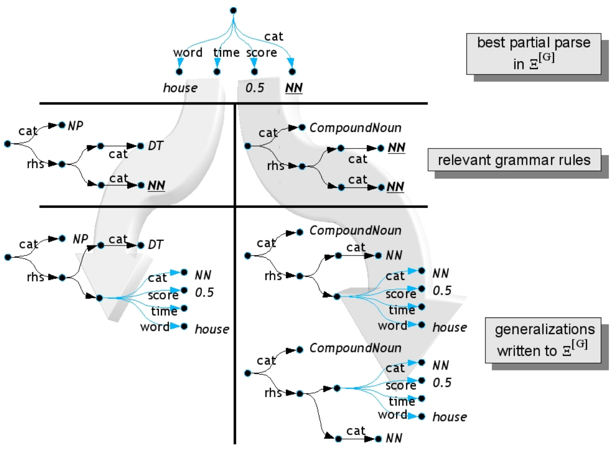


Figure 5: Example of GENERALISATION.

2.4 Specification

The SPECIFIER thread provides the impetus towards complete coverage of the input, as simplified in Algorithm 2 (see Figure 6). It combines parses in its subspaces that have the same top-level grammar expansion but different instantiated constituents. The resulting parse merges the semantics of the two original graphs only if unification succeeds, providing a hard constraint against the combination of incongruous information. The result, Ψ, of specification *must* be written to $\Xi^{[G]}$, otherwise Ψ could never appear on the RHS of another partial parse. We show how associated vulnerabilities are overcome in §3.2 and §3.4.

Specification is commutative and will always provide more information than its constituent graphs if it does not fail, unlike the 'overlay' method of SMARTKOM (Alexandersson and Becker, 2001), which basically provides a subsumption mechanism over background knowledge.

Algorithm 2: Simplified Specification

```
Data: Subspaces $\Xi^{[SAct]}$ and $\Xi^{[SInact]}$
while data remains in $\Xi^{[SAct]}$ do
    $\Psi_s$ := highest scoring graph in $\Xi^{[SAct]}$
    $\Psi_j$ := highest scoring graph in $\Xi^{[SInact]}$
    s.t. Cat $(\Psi_j) =$ Cat $(\Psi_s)$
    while $\exists\Psi_j$ do
        $\Psi_i$ := Unify $(\Psi_s, \Psi_j)$
        if $\exists\Psi_i$ then
            Apply Score $(\Psi_i)$ to $\Psi_i$
            Insert $\Psi_i$ into $\Xi^{[G]}$
        $\Psi_j$ := next highest scoring graph from
        $\Xi^{[SInact]}$ s.t. Cat $(\Psi_j) =$ Cat $(\Psi_s)$
        ; // Optionally stop after $I$
        iterations, for some $I$
    Move $\Psi_s$ into $\Xi^{[SInact]}$
```

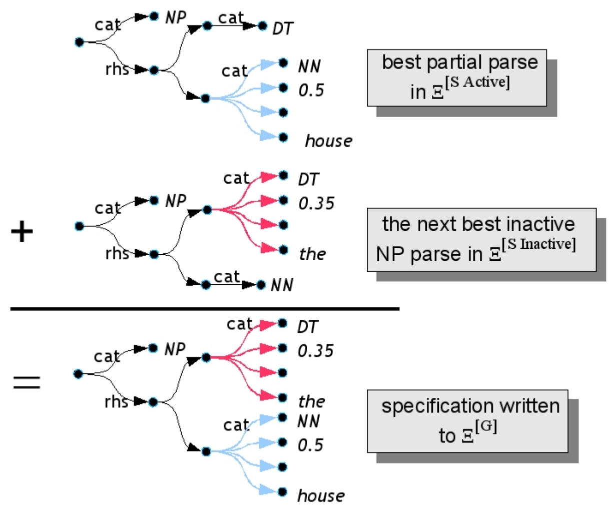

cat
NP
cat
DT
rhs
cat
NN
0.5
house
best partial parse in Ξ[S Active]
+
cat
DT
cat
NP
0.35
rhs
the
cat
NN
the next best inactive NP parse in Ξ[S Inactive]
=
cat
DT
cat
NP
0.35
rhs
the
NN
cat
0.5
house
specification written to Ξ[G]

Figure 6: Example of SPECIFICATION.

2.5 Cognition

The COGNITION thread monitors the best sentence-level hypothesis, Ψ_B, in $\Xi^{[SInact]}$, and terminates the search process once Ψ_B has remained unchallenged by new competing parses for some period of time.

Once found, COGNITION communicates Ψ_B to the APPLICATION. Both COGNITION and the APPLICATION read state information from the MySQL WORLD database, as discussed in §3.5, though only the latter can modify it.

3 Applying Domain-Centric Knowledge

Upon being created, all partial parses are assigned a score approximating its likelihood of being part of an accepted multimodal sentence. The score of partial parse Ψ, $\text{SCORE}(\Psi) = \sum_{i=0}^{|S|} \omega_i \kappa_i(\Psi)$, is a weighted linear combination of independent scoring modules (KNOWLEDGE SOURCES). Each module presents a score function $\kappa_i : \Psi \to \Re_{[0..1]}$ according to a unique criterion of multimodal language, weighted by ω_i, also on $\Re_{[0..1]}$. Some modules provide 'hard constraints' that can outright forbid unification, returning $\kappa_i = -\infty$ in those cases. A subset of the criteria we have explored are outlined below.

3.1 Temporal Alignment (κ_1)

By modelling the timespans of parses as Gaussians, where μ and σ are determined by the midpoint and $\frac{1}{2}$ the distance between the two endpoints, respectively - we can promote parses whose constituents are closely related in time with the *symmetric Kullback-Leibler divergence*, $D_{KL}(\Psi_1, \Psi_2) = \frac{(\sigma_1^2-\sigma_2^2)^2+((\mu_1-\mu_2)(\sigma_1^2+\sigma_2^2))^2}{4\sigma_1^2\sigma_2^2}$. Therefore, κ_1 promotes more locally-structured parses, and co-occuring multimodal utterances.

3.2 Ancestry Constraint (κ_2)

A consequence of accepting n-best lexical hypotheses for each word is that we risk unifying parses that include two competing hypotheses. For example, if our speech TRACKER produces hypotheses "*horse*" and "*house*" for ambiguous input, then κ_2 explicitly prohibits the parse "*the <u>horse</u> and the <u>house</u>*" with flags on lexical content.

3.3 Probabilistic Grammars (κ_3)

We emphasise more common grammatical constructions by augmenting each grammar rule with an associated probability, $P(\Gamma_i)$, and assigning $\kappa_3(\Psi) = P(\text{RULE}(\Psi)) \cdot \prod_{\Psi_c=\text{constituent of }\Psi} \kappa_3(\Psi_c)$ where RULE is the top-level expansion of Ψ.

Probabilities are trainable by maximum likelihood estimation on annotated data. Within the context of CLAVIUS , κ_3 promotes the processing of new input words and shallower parse trees.

3.4 Information Content (κ_4), Coverage (κ_5)

The κ_4 module partially orders parses by preferring those that maximise the joint entropy between the semantic variables of its constituent parses. Furthermore, we use a shifted sigmoid $\kappa_5(\Psi) = \frac{2}{1+e^{-\frac{2}{5}\textsc{NumWordsIn}(\Psi)}} - 1$, to promote parses that maximise the number of 'words' in a parse. These two modules together are vital in choosing fully specified sentences.

3.5 Functional Constraints (κ_6)

Each grammar rule Γ_i can include constraint functions $f : \Psi \rightarrow \Re_{[0,1]}$ parametrised by values in instantiated graphs. For example, the function $\textsc{T\_follows}(\Psi_1, \Psi_2)$ returns 1 if constituent Ψ_2 follows Ψ_1 in time, and $-\infty$ otherwise, thus maintaining ordering constraints. Functions are dynamically loaded and executed during scoring.

Since functions are embedded directly within parse graphs, their return values can be directly incorporated into those parses, allowing us to utilise data in the WORLD. For example, the function $\textsc{ObjectAt}(x, y, \&o)$ determines if an object exists at point (x, y), as determined by a pointing gesture, and writes the type of this object, o, to the graph, which can later further constrain the search.

4 Early Results

We have constructed a simple blocks-world experiment where a user can move, colour, create, and delete geometric objects using speech and pointing gestures with 74 grammar rules, 25 grammatical categories, and a 43-word vocabulary. Ten users were recorded interacting with this system, for a combined total of 2.5 hours of speech and gesture data, and 2304 multimodal utterances. Our randomised data collection mechanism was designed to equitably explore the four command types. Test subjects were given no indication as to the types of phrases we expected - but were rather shown a collection of objects and were asked to replicate it, given the four basic types of actions.

Several aspects of the parser have been tested at this stage and are summarised below.

4.1 Accuracy

Table 1 shows three hand-tuned configurations of the module weights ω_i, with $\omega_2 = 0.0$, since κ_2 provides a 'hard constraint' (§3.2).

Figure 7 shows sentence-level precision achieved for each Ω_i on each of the four tasks, where precision is defined as the proportion of correctly executed sentences. These are compared against the CMU Sphinx-4 speech recogniser using the unimodal projection of the multimodal grammar. Here, conjunctive phrases such as *"Put a sphere here and colour it yellow"* are classified according to their first clause.

Presently, correlating the coverage and probabilistic grammar constraints with higher weights ($> 30\%$) appears to provide the best results. Creation and colouring tasks appeared to suffer most due to missing or misunderstood head-noun modifiers (ie., object colour). In these examples, CLAVIUS ranged from a -51.7% to a 62.5% relative error reduction rate over all tasks.

Config	ω_1	$\omega_2^{(*)}$	ω_3	ω_4	ω_5	ω_6
Ω_1	0.4	0.0	0.3	0.1	0.1	0.1
Ω_2	0.2	0.0	0.1	0.3	0.2	0.2
Ω_3	0.1	0.0	0.3	0.3	0.15	0.15

Table 1: Three weight configurations.

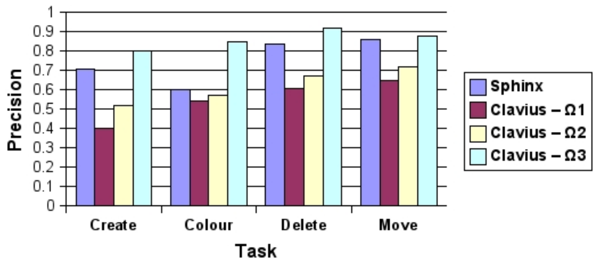


Figure 7: Precision across the test tasks.

4.2 Work Expenditure

To test whether the best-first approach compensates for CLAVIUS ' looser constraints (§1.2), a simple bottom-up multichart parser (§1.1) was constructed and the average number of edges it produces on sentences of varying length was measured. Figure 8 compares this against the average number of edges produced by CLAVIUS on the same data. In particular, although CLAVIUS generally finds the parse it will accept relatively quickly ('CLAVIUS - found'), the COGNITION module will delay its acceptance ('CLAVIUS - accepted') for a time. Further tuning will hopefully reduce this 'waiting period'.

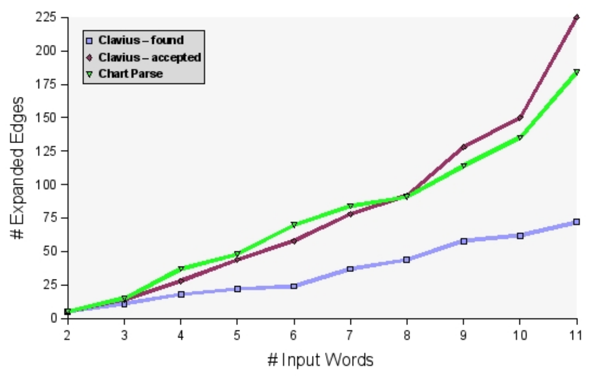


Figure 8: Number of edges expanded, given sentence length.

5 Remarks

CLAVIUS consistently ignores over 92% of dysfluencies (eg. "uh") and significant noise events in tracking, apparently as a result of the partial qualifications discussed in §1.2.3, which is especially relevant in noisy environments. Early unquantified observation also suggests that a result of unordered constituents is that parses incorporating lead words - head nouns, command verbs and pointing gestures in particular - are emphasised and form sentence-level parses early, and are later 'filled in' with function words.

5.1 Ongoing Work

There are at least four avenues open to exploration in the near future. First, applying the parser to directed two-party dialogue will explore context-sensitivity and a more complex grammar. Second, the architecture lends itself to further parallelism - specifically by permitting $P > 1$ concurrent processing units to dynamically decide whether to employ the GENERALISER or SPECIFIER, based on the sizes of shared active subspaces.

We are also currently working on scoring modules that incorporate language modelling (with discriminative training), and prosody-based co-analysis. Finally, we have already begun work on automatic methods to train scoring parameters, including the distribution of ω_i, and module-specific training.

6 Acknowledgements

Funding has been provided by la bourse de maitrisse of the fonds québécois de la recherche sur la nature et les technologies.

References

Ageno, A., Rodriguez, H. 2000 *Extending Bidirectional Chart Parsing with a Stochastic Model*, in Proc. of TSD 2000, Brno, Czech Republic.

Alexandersson, J. and Becker, T. 2001 *Overlay as the Basic Operation for Discourse Processing in a Multimodal Dialogue System* in Proc. of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Seattle, WA.

Bolt, R.A. 1980 *"Put-that-there": Voice and gesture at the graphics interface* in Proc. of SIGGRAPH 80 ACM Press, New York, NY.

Boussemart, Y., Rioux, F., Rudzicz, F., Wozniewski, M., Cooperstock, J. 2004 *A Framework for 3D Visualisation and Manipulation in an Immersive Space using an Untethered Bimanual Gestural Interface* in Proc. of VRST 2004 ACM Press, Hong Kong.

Dowding, J. et al. 1993 *Gemini: A Natural Language System For Spoken-Language Understanding* in Meeting of the ACL, ACL, Morristown, NJ.

Holzapfel, H., Nickel, K., Stiefelhagen, R. 2004 *Implementation and evaluation of a constraint-based multimodal fusion system for speech and 3D pointing gestures*, in ICMI '04: Proc. of the 6th intl. conference on Multimodal interfaces, ACM Press, New York, NY.

Johnston, M. 1998 *Unification-based multimodal parsing*, in Proc. of the 36th annual meeting of the ACL, ACL, Morristown, NJ.

Johnston, M., Bangalore, S. 2000 *Finite-state multimodal parsing and understanding* in Proc. of the 18th conference on Computational linguistics ACL, Morristown, NJ.

Kettebekov, S., et al. 2002 *Prosody Based Co-analysis of Deictic Gestures and Speech in Weather Narration Broadcast*, in Workshop on Multimodal Resources and Multimodal System Evaluation. (LREC 2002), Las Palmas, Spain.

McNeill, D. 1992 *Hand and mind: What gestures reveal about thought* University of Chicago Press and CSLI Publications, Chicago, IL.

Rocio, V., Lopes, J.G. 1998 *Partial Parsing, Deduction and Tabling* in TAPD 98

Tomita, M. 1985 *An Efficient Context-Free Parsing Algorithm for Natural Languages*, in Proc. Ninth Intl. Joint Conf. on Artificial Intelligence, Los Angeles, CA.

Author Index