ACL-05

# Interactive Poster and Demonstration Sessions

## Proceedings

26 June 2005
University of Michigan

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

# Introduction

The ACL 2005 Interactive Poster and Demonstration session took place on Sunday, June 26, 2005 in Ann Arbor Michigan, on the first day of the 43rd Annual Meeting of the Association for Computational Linguistics. There were 56 submissions to this event, of which 31 were selected for presentation, resulting in a 55% acceptance rate.

Our goal this year was to have a program made up of implemented systems that reflect previously unpublished work (Interactive Posters) or previously published but subsequently improved existing systems (Demonstrations). Thus, while we maintained a fairly traditional definition of *Demonstration*, we introduced an experimental sense of *Interactive Poster* that required that they present novel and previously unpublished ideas *and* that they be supported by an implemented system.

Thus, the criteria for acceptance of an Interactive Poster was the novel scientific contributions of the work, and the effectiveness of the implemented system and (optional) more traditional poster in making those points. There were 37 Interactive Posters submitted, of which 18 were accepted.

Demonstrations were expected to highlight mature systems or prototypes that show how NLP technologies are used to solve practically important problems. The criteria for acceptance of a Demonstration was that it must already be described in the published literature in sufficient detail to allow replication, or the published paper should provide this level of detail. In addition, the Demonstration should address an application of broad interest in such a way that it can be appreciated by the diverse audience that attends ACL. There were 19 Demonstrations submitted, of which 13 were accepted.

First and foremost, we would like to thank the General Conference Chair of ACL 2005, Kevin Knight, for giving us the flexibility and support to pursue a somewhat non–traditional formulation of this event. We would also like to thank the members of the Program Committee for their speedy and insightful reviews, and for their help in refining our definitions of Interactive Poster and Demonstration. Finally, we owe great thanks to Dragomir Radev, who worked tirelessly as the Local Arrangements Chair to accommodate a wide range of requests that we made regarding both the scheduling and physical facilities for this event. We are deeply grateful to him for his hard work which certainly helped to make this a very successful event.

Masaaki Nagata and Ted Pedersen
Co–Chairs

# Organizers

**Co–Chairs:**

Masaaki Nagata (NTT Cyber Space Laboratories)
Ted Pedersen (University of Minnesota, Duluth)

**Program Committee:**

John Atkinson (Universidad de Concepcion)
Jill Burstein (Educational Testing Service)
Mike Calcagno (Microsoft Speech and Natural Language)
Charles Callaway (University of Edinburgh)
Ciprian Chelba (Microsoft Research)
Silviu Cucerzan (Microsoft Research)
Mark Dras (Macquarie University)
Gregor Erbach (DFKI GmbH)
Reva Freedman (Northern Illinois University)
Jianfeng Gao (Microsoft Research Asia)
Alexander Gelbukh (National Polytechnic Institute, Mexico)
Alexander Hauptmann (Carnegie Mellon University)
Yoshihiko Hayashi (Osaka University)
Graeme Hirst (University of Toronto)
Camelia Ignat (European Commission - Joint Research Centre)
Diana Inkpen (University of Ottawa)
Su Jian (Institute for Infocomm Research)
Jin-Dong Kim (University of Tokyo)
Taku Kudo (Google)
Sadao Kurohashi (University of Tokyo)
Claudia Leacock (Pearson Knowledge Technologies)
Gary Geunbae Lee (Pohang University of Science and Technology)
Diane Litman (University of Pittsburgh)
Sun Maosong (Tsinghua University)
Rada Mihalcea (University of North Texas)
Vibhu Mittal (Google/Carnegie Mellon University)
Mikio Nakano (Honda Research Institute Japan)
Manabu Okumura (Tokyo Institute of Technology)
Serguei Pakhomov (Mayo College of Medicine)
Patrick Pantel (USC Information Sciences Institute)
Marius Pasca (Google)
David Pierce (University at Buffalo)
Bruno Pouliquen (European Commission - Joint Research Centre)
Philip Resnik (University of Maryland)
Carolyn Penstein Rosé (Carnegie Mellon University)
Horacio Saggion (University of Sheffield)

William Schuler (University of Minnesota, Twin Cities)
Kiyoaki Shirai (Japan Advanced Institute of Science and Technology)
Thamar Solorio (Instituto Nacional de Astrofísica, Óptica, y Electrónica)
Virach Sornlertlamvanich (NICT Asia Research Center)
Ralf Steinberger (European Commission - Joint Research Centre)
Keh-Yih Su (Behavior Design Corporation)
Peter Turney (National Research Council of Canada)
Masao Utiyama (NICT)
Takehito Utsuro (Kyoto University)
Janyce Wiebe (University of Pittsburgh)
Peter Wiemer-Hastings (DePaul University)

**Additional Reviewers:**

John Chen (Microsoft Research Asia)
Hal Daume III (USC Information Sciences Institute)
Alexander Fraser (USC Information Sciences Institute)
Zhang Jie (Institute for Infocomm Research)
Mu Li (Microsoft Research Asia)
Yajuan Lv (Microsoft Research Asia)
Liang Zhou (USC Information Sciences Institute)

# Table of Contents

## Interactive Poster (IP) and Demonstration (D) Program

**Sunday, June 26, 2005 (10:00 am – 12:00 noon)**

**Sunday, June 26, 2005 (1:30 pm – 3:30 pm)**

# An Information-State Approach to Collaborative Reference

**David DeVault**[1] **Natalia Kariaeva**[2] **Anubha Kothari**[2] **Iris Oved**[3] and **Matthew Stone**[1]
[1]Computer Science [2]Linguistics [3]Philosophy and Center for Cognitive Science
Rutgers University
Piscataway NJ 08845-8020
`Firstname.Lastname@Rutgers.Edu`

**Abstract**

We describe a dialogue system that works with its interlocutor to identify objects. Our contributions include a concise, modular architecture with reversible processes of understanding and generation, an information-state model of reference, and flexible links between semantics and collaborative problem solving.

## 1 Introduction

People work together to make sure they understand one another. For example, when identifying an object, speakers are prepared to give many alternative descriptions, and listeners not only show whether they understand each description but often help the speaker find one they do understand (Clark and Wilkes-Gibbs, 1986). This natural collaboration is part of what makes human communication so robust to failure. We aim both to explain this ability and to reproduce it.

In this paper, we present a novel model of collaboration in referential linguistic communication, and we describe and illustrate its implementation. As we argue in Section 2, our approach is unique in combining a concise abstraction of the dynamics of joint activity with a reversible grammar-driven model of referential language. In the new information-state model of reference we present in Section 3, interlocutors work together over multiple turns to associate an entity with an agreed set of concepts that characterize it. On our approach, utterance planning and understanding involves reasoning about how domain-independent linguistic forms can be used in context to contribute to the task; see Section 4. Our system reduces to four modules: understanding, update, deliberation and generation, together with some supporting infrastructure; see Section 5. This design derives the efficiency and flexibility of referential communication from carefully-designed representation and reasoning in this simple architecture; see Section 6. With this proof-of-concept implementation, then, we provide a jumping-off point for more detailed investigation of knowledge and processes in conversation.

## 2 Overview and Related Work

Our demonstration system plays a referential communication game, much like the one that pairs of human subjects play in the experiments of Clark and Wilkes-Gibbs (1986). We describe each episode in this game as an activity involving the coordinated action of two participants: a *director* $D$ who knows the referent $R$ of a target variable $T$ and a *matcher* $M$ whose task is to identify $R$. Our system can play either role, $D$ or $M$, using virtual objects in a graphical display as candidate targets and distractors, and using text as its input and output. Our system uses the same task knowledge and the same grammar whichever role it plays. Of course, the system also draws on private knowledge to decide how best to carry out its role; for now it describes objects using the domain-specific iteration proposed by Dale and Reiter (1995). The knowledge we have formalized is targeted to a proof-of-concept implementation, but we see no methodological obstacle in adding to the system's resources.

We exemplify what our system does in (1).

(1) a. S: This one is a square.
  b. U: Um-hm...
  c. S: It's light brown.
  d. U: You mean like tan?
  e. S: Yeah.
  f. S: It's solid.
  g. U: Got it.

The system (S) and user (U) exchange seven utterances in the course of identifying a tan solid square.

We achieve this interaction using the information-state approach to dialogue system design (Larsson and Traum, 2000). This approach describes dialogue as a coordinated effort to maintain an agreed record of the state of the conversation. Our model contrasts with traditional plan-based models, as exemplified by Heeman and Hirst's model of goals and beliefs in collaborative reference (1995). Our approach abstracts away from such details of individuals' mental states and cognitive processes, for principled reasons (Stone, 2004a). We are able to capture these details *implicitly* in the dynamics of conversation, whereas plan-based models must represent them *explicitly*. Our representations are simpler than Heeman and Hirst's but support more flexible dialogue. For example, their approach to (1) would have interlocutors coordinating on goals and beliefs about a syntactic representation for *the tan solid square*; for us, this description and the interlocutors' commitment to it are abstract results of the underlying collaborative activity.

Another important antecedent to our work is Purver's (2004) characterization of clarification of names for objects and properties. We extend this work to develop a treatment of referential descriptive clarification. When we describe things, our descriptions grow incrementally and can specify as much detail as needed. Clarification becomes correspondingly cumulative and open-ended. Our revised information state includes a model of cumulative and open-ended collaborative activity, similar to that advocated by Rich et al. (2001). We also benefit from a reversible goal-directed perspective on descriptive language (Stone et al., 2003).

## 3 Information State

Our information state (IS) models the ongoing collaboration using a stack of tasks. For a task of collaborative reference, the IS tracks how interlocutors together set up and solve a constraint-satisfaction problem to identify a target object. In any state, $D$ and $M$ have agreed on a target variable $T$ and a set of constraints that the value of $T$ must satisfy. When $M$ recognizes that these constraints identify $R$, the task ends successfully. Until then, $D$ can take actions that contribute new constraints on $R$. Importantly, what $D$ says adds to what is already known about $R$, so that the identification of $R$ can be accomplished across multiple sentences with heterogeneous syntactic structure.

Our IS also allows subtasks of questioning or clarification that interlocutors can use to maintain alignment. The same constraint-satisfaction model is used not only for referring to displayed objects but also for referring to abstract entities, such as actions or properties. Our IS tracks the salience of entity and property referents and, like Purver's, maintains the previous utterance for reference in clarification questions. Note, however, that we do not factor updates to the IS through an abstract taxonomy of speech acts. Instead, utterances directly make domain moves, such as adding a constraint, so our architecture allows utterances to trigger an open-ended range of domain-specific updates.

## 4 Linguistic Representations

The way utterances signal task contributions is through a collection of presupposed constraints. To understand an utterance, we solve the utterance's grammatically-specified semantic constraints. An interpretation is only feasible if it represents a contextually-appropriate contribution to the ongoing task. Symmetrically, to generate an utterance, we use the grammar to formulate a set of constraints; these constraints must identify the contribution the system intends to make. We view interpreted linguistic structures as representing communicative intentions; see (Stone et al., 2003) or (Stone, 2004b).

As in (DeVault et al., 2004), a *knowledge interface* mediates between domain-general meanings and the domain-specific ontology supported in a particular application. This allows us to build interpretations using domain-specific representations for referents, for task moves, and for the domain properties that characterize referents.

## 5 Architecture

Our system is implemented in Java. A set of interface types describes the flow of information and control through the architecture. The representation and reasoning outlined in Sections 3 and 4 is accomplished by implementations of these interfaces that realize our approach. Modules in the architecture exchange messages about events and their interpretations. (1) Deliberation responds to changes in the IS by proposing task moves. (2) Generation constructs collaborative intentions to accomplish the planned task moves. (3) Understanding infers collaborative intentions behind user actions. Generation and understanding share code to construct intentions for utterances, and both carry out a form of inference to the best explanation. (4) Update advances the IS symmetrically in response to intentions signaled by the system or recognized from the user; the symmetric architecture frees the designer from programming complementary updates in a symmetrical way. Additional supporting infrastructure handles the recognition of input actions, the realization of output actions, and interfacing between domain knowledge and linguistic resources.

Our system is designed not just for users to interact with, but also for demonstrating and debugging the system's underlying models. Processing can be paused at any point to allow inspection of the system's representations using a range of visualization tools. You can interactively explore the IS, including the present state of the world, the agreed direction of the ongoing task, and the representation of linguistic distinctions in salience and information status. You can test the grammar and other interpretive resources. And you can visualize the search space for understanding and generation.

## 6 Example

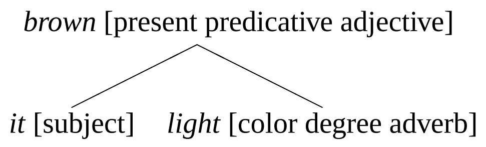
Let us return to dialogue (1). Here the system represents its moves as successively constraining the shape, color and pattern of the target object. In generating (1c), the system iteratively elaborates its description from *brown* to *light brown* in an attempt to identify the object's color unambiguously. The user's clarification request at (1d) marks this description of color as problematic and so triggers a nested instance of the collaborative reference task. At (1e) the system adds the user's proposed constraint and (we assume) solves this nested subtask. The system returns to the main task at (1f) having grounded the color constraint and continues by identifying the pattern of the target object.

Let us explore utterance (1c) in more detail. The IS records the status of the identification process. The system is the director; the user is the matcher. The target is represented provisionally by a discourse referent $t_1$, and what has been agreed so far is that the current target is a square of the relevant sort for this task, represented in the agent as $\textit{square-figure-object}(t_1)$. In addition, the system has privately recorded that square $o_1$ is the referent it must identify. For this IS, it is expected that the director will propose an additional constraint identifying $t_1$. The discourse state represents $t_1$ as being *in-focus*, or available for pronominal reference.

Deliberation now gives the generator a specific move for the system to achieve:

(2) $\textit{add-constraint}(t_1, \textit{color-sandybrown}(t_1))$

The content of the move in (2) is that the system should update the collaborative reference task to include the constraint that the target is drawn in a particular, domain-specific color (RGB value F4-A4-60, or XHTML standard "sandy brown"). The system finds an utterance that achieves this by exploring head-first derivations in its grammar; it arrives at the derivation of *it's light brown* in (3).

(3)

*brown* [present predicative adjective]
*it* [subject]
*light* [color degree adverb]

A set of presuppositions connect this linguistic structure to a task domain; they are given in (4a). The relevant instances in this task are shown in (4b).

(4) a. $\textit{predication}(M) \wedge \textit{brown}(C) \wedge \textit{light}(C)$

b. $\textit{predication}(\textit{add-constraint}) \wedge$
$\textit{brown}(\textit{color-sandybrown}) \wedge$
$\textit{light}(\textit{color-sandybrown})$

The utterance also uses *it* to describe a referent $X$ so presupposes that *in-focus*$(X)$ holds. The move effected by the utterance is schematized as $M(X,C(X))$. Given the range of possible task moves in the current context, the constraints specified by the grammar for (3) are modeled as determining the instantiation in (2). The system realizes the utterance and assumes, provisionally, that the utterance achieves its intended effect and records the new constraint on $t_1$.

Because the generation process incorporates entirely declarative reasoning, it is normally reversible. Normally, the interlocutor would be able to identify the speaker's intended derivation, associate it with the same semantic constraints, resolve those constraints to the intended instances, and thereby discover the intended task move. In our example, this is not what happens. Recognition of the user's clarification request is triggered as in (Purver, 2004). The system fails to interpret utterance (1d) as an appropriate move in the main reference task. As an alternative, the system "downdates" the context to record the fact that the system's intended move may be the subject of explicit grounding. This involves pushing a new collaborative reference task on the stack of ongoing activities. The system remains the director, the new target is the variable $C$ in interpretation and the referent to be identified is the property *color-sandybrown*. Interpretation of (1d) now succeeds.

## 7 Discussion

Our work bridges research on collaborative dialogue in AI (Rich et al., 2001) and research on pragmatics in computational linguistics (Stone et al., 2003). The two traditions have a lot to gain from reconciling their assumptions, if as Clark (1996) suggests, people's language use is coextensive with their joint activity. There are implications both ways.

For pragmatics, our model suggests that language use requires collaboration in part because reaching agreement about content involves substantive social knowledge and coordination. Indeed, we suspect that collaborative reference is only one of many relevant social processes. For collaborative dialogue systems, adopting rich declarative linguistic representations enables us to directly interface the core modules of a collaborative system with one another. In language understanding, for example, we can collapse together notional subprocesses like semantic reconstruction, reference resolution, and intention recognition and solve them in a uniform way.

Our declarative, reversible approach supports an analysis of how the system's specifications drive its input-output behavior. The architecture of this system thus provides the groundwork for further investigations into the interaction of social, linguistic, cognitive and even perceptual and developmental processes in meaningful communication.

## Acknowledgements

Supported in part by NSF HLC 0308121. Thanks to Paul Tepper.

## References

H. H. Clark and D. Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22:1–39.

H. H. Clark. 1996. *Using Language*. Cambridge.

R. Dale and E. Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 18:233–263.

D. DeVault, C. Rich, and C. L. Sidner. 2004. Natural language generation and discourse context: Computing distractor sets from the focus stack. In *FLAIRS*.

P. Heeman and G. Hirst. 1995. Collaborating on referring expressions. *Comp. Ling.*, 21(3):351–382.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Eng.*, 6:323–340.

M. Purver. 2004. *The Theory and Use of Clarification Requests in Dialogue*. Ph.D. thesis, Univ. of London.

C. Rich, C. L. Sidner, and N. Lesh. 2001. COLLAGEN: applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22:15–25.

M. Stone, C. Doran, B. Webber, T. Bleam, and M. Palmer. 2003. Microplanning with communicative intentions. *Comp. Intelligence*, 19(4):311–381.

M. Stone. 2004a. Communicative intentions and conversational processes. In J. Trueswell and M. K. Tanenhaus, editors, *Approaches to Studying World-Situated Language Use*, pages 39–70. MIT.

M. Stone. 2004b. Intention, interpretation and the computational structure of language. *Cognitive Science*, 28(5):781–809.

# Accessing GermaNet Data and Computing Semantic Relatedness

**Iryna Gurevych and Hendrik Niederlich**
EML Research gGmbH
Schloss-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
`http://www.eml-research.de/∼gurevych`

## Abstract

We present an API developed to access GermaNet, a lexical semantic database for German represented in XML. The API provides a set of software functions for parsing and retrieving information from GermaNet. Then, we present a case study which builds upon the GermaNet API and implements an application for computing semantic relatedness according to five different metrics. The package can, again, serve as a software library to be deployed in natural language processing applications. A graphical user interface allows to interactively experiment with the system.

## 1 Motivation

The knowledge encoded in WordNet (Fellbaum, 1998) has proved valuable in many natural language processing (NLP) applications. One particular way to integrate semantic knowledge into applications is to compute semantic similarity of WordNet concepts. This can be used e.g. to perform word sense disambiguation (Patwardhan et al., 2003), to find predominant word senses in untagged text (McCarthy et al., 2004), to automatically generate spoken dialogue summaries (Gurevych & Strube, 2004), and to perform spelling correction (Hirst & Budanitsky, 2005).

Extensive research concerning the integration of semantic knowledge into NLP for the English language has been arguably fostered by the emergence of WordNet::Similarity package (Pedersen et al., 2004).[1] In its turn, the development of the WordNet based semantic similarity software has been facilitated by the availability of tools to easily retrieve data from WordNet, e.g. WordNet::QueryData,[2] jwnl.[3]

[1]http://www.d.umn.edu/∼tpederse/similarity.html

Research integrating semantic knowledge into NLP for languages other than English is scarce. On the one hand, there are fewer computational knowledge resources like dictionaries, broad enough in coverage to be integrated in robust NLP applications. On the other hand, there is little off-the-shelf software that allows to develop applications utilizing semantic knowledge from scratch. While WordNet counterparts do exist for many languages, e.g. GermaNet (Kunze & Lemnitzer, 2002) and EuroWordNet (Vossen, 1999), they differ from WordNet in certain design aspects. E.g. GermaNet features non-lexicalized, so called *artificial* concepts that are non-existent in WordNet. Also, the adjectives are structured hierarchically which is not the case in WordNet. These and other structural differences led to divergences in the data model. Therefore, WordNet based implementations are not applicable to GermaNet. Also, there is generally lack of experimental evidence concerning the portability of e.g. WordNet based semantic similarity metrics to other wordnets and their sensitivity to specific factors, such as network structure, language, etc. Thus, for a researcher who wants to build a semantic relatedness application for a language other than English, it is difficult to assess the effort and challenges involved in that.

Departing from that, we present an API which allows to parse and retrieve data from GermaNet. Though it was developed following the guidelines for creating WordNet, GermaNet features a couple of divergent design decisions, such as e.g. the use of non-lexicalized concepts, the association relation between synsets and the small number of textual definitions of word senses. Furthermore, we

[2]http://search.cpan.org/dist/WordNet-QueryData
[3]http://sourceforge.net/projects/jwordnet

build an application accessing the knowledge in GermaNet and computing semantic relatedness of GermaNet word senses according to five different metrics. Three of these metrics have been adapted from experiments on English with WordNet, while the remaining two are based on automatically generated definitions of word senses and were developed in the context of work with GermaNet.

## 2 GermaNet API

The API for accessing GermaNet has to provide functions similar to the API developed for WordNet. We evaluated the C-library distributed together with GermaNet V4.0 and the XML encoded version of GermaNet (Lemnitzer & Kunze, 2002). As we wanted the code to be portable across platforms, we built upon the latter. The XML version of GermaNet is parsed with the help of the Apache Xerces parser, http://xml.apache.org/ to create a JAVA object representing GermaNet. For stemming the words, we use the functionality provided by the Porter stemmer for the German language, freely available from http://snowball.tartarus.org/german/stemmer.html. Thus, the GermaNet object exists in two versions, the original one, where the information can be accessed using words, and the stemmed one, where the information can be accessed using word stems.

We implemented a range of JAVA based methods for querying the data. These methods are organized around the notions of word sense and synset. On the word sense (WS) level, we have the following methods: *getAntonyms()* retrieves all antonyms of a given WS; *getArtificial()* indicates whether a WS is an artificial concept; *getGrapheme()* gets a graphemic representation of a WS; *getParticipleOf()* retrieves the WS of the verb that the word sense is a participle of; *getPartOfSpeech()* gets the part of speech associated with a WS; *getPertonym()* gives the WS that the word sense is derived from; *getProperName()* indicates whether the WS is a proper name; *getSense()* yields the sense number of a WS in GermaNet; *getStyle()* indicates if the WS is stylistically marked; *getSynset()* returns the corresponding synset; *toString()* yields a string representing a WS.

On the synset level, the following information can be accessed: *getAssociations()* returns all associations; *getCausations()* gets the effects that a given synset is a cause of; *getEntailments()* yields synsets that entail a given synset; *getHolonyms()*, *getHyponyms()*, *getHypernyms()*, *getMeronyms()* return a list of holonyms, hyponyms, immediate hypernyms, and meronyms respectively; *getPartOfSpeech()* returns the part of speech associated with word senses of a synset; *getWordSenses()* returns all word senses constituting the synset; *toString()* yields a string representation of a synset.

The metrics of semantic relatedness are designed to employ this API. They are implemented as classes which use the API methods on an instance of the GermaNet object.

## 3 Semantic Relatedness Software

In GermaNet, nouns, verbs and adjectives are structured within hierarchies of *is-a* relations.[4] GermaNet also contains information on additional lexical and semantic relations, e.g. hypernymy, meronymy, antonymy, etc. (Kunze & Lemnitzer, 2002). A semantic relatedness metric specifies to what degree the meanings of two words are related to each other. E.g. the meanings of *Glas* (Engl. *glass*) and *Becher* (Engl. *cup*) will be typically classified as being closely related to each other, while the relation between *Glas* and *Juwel* (Engl. *gem*) is more distant. *RelatednessComparator* is a class which takes two words as input and returns a numeric value indicating semantic relatedness for the two words. Semantic relatedness metrics have been implemented as descendants of this class.

Three of the metrics for computing semantic relatedness are information content based (Resnik, 1995; Jiang & Conrath, 1997; Lin, 1998) and are also implemented in WordNet::Similarity package. However, some aspects in the normalization of their results and the task definition according to which the evaluation is conducted have been changed (Gurevych & Niederlich, 2005). The metrics are implemented as classes derived from *InformationBasedComparator*, which is in its turn derived from the class *PathBasedComparator*. They make use of both the GermaNet hierarchy and statistical corpus evidence, i.e. information content.

[4]As mentioned before, GermaNet abandoned the cluster-approach taken in WordNet to group adjectives. Instead a hierarchical structuring based on the work by Hundsnurscher & Splett (1982) applies, as is the case with nouns and verbs.

We implemented a set of utilities for computing information content of German word senses from German corpora according to the method by Resnik (1995). The TreeTagger (Schmid, 1997) is employed to compile a part-of-speech tagged word frequency list. The information content values of GermaNet synsets are saved in a text file called an information content map. We experimented with different configurations of the system, one of which involved stemming of corpora and the other did not involve any morphological processing. Contrary to our intuition, there was almost no difference in the information content maps arising from the both system configurations, with and without morphological processing. Therefore, the use of stemming in computing information content of German synsets seems to be unjustified.

The remaining two metrics of semantic relatedness are based on the Lesk algorithm (Lesk, 1986). The Lesk algorithm computes the number of overlaps in the definitions of words, which are sometimes extended with the definitions of words related to the given word senses (Patwardhan et al., 2003). This algorithm for computing semantic relatedness is very attractive. It is conceptually simple and does not require an additional effort of corpus analysis compared with information content based metrics.

However, a straightforward adaptation of the Lesk metric to GermaNet turned out to be impossible. Textual definitions of word senses in GermaNet are fairly short and small in number. In cotrast to WordNet, GermaNet cannot be employed as a machine-readable dictionary, but is primarily a conceptual network. In order to deal with this, we developed a novel methodology which generates definitions of word senses automatically from GermaNet using the GermaNet API. Examples of such automatically generated definitions can be found in Gurevych & Niederlich (2005). The method is implemented in the class *PseudoGlossGenerator* of our software, which automatically generates glosses on the basis of the conceptual hierarchy.

Two metrics of semantic relatedness are, then, based on the application of the Lesk algorithm to definitions, generated automatically according to two system configurations. The generated definitions can be tailored to the task at hand according to a set of parameters defining which related concepts have to be included in the final definition. Experiments carried out to determine the most effective parameters for generating the definitions and employing those to compute semantic relatedness is described in Gurevych (2005). Gurevych & Niederlich (2005) present a description of the evaluation procedure for five implemented semantic relatedness metrics against a human *Gold Standard* and the evaluation results.

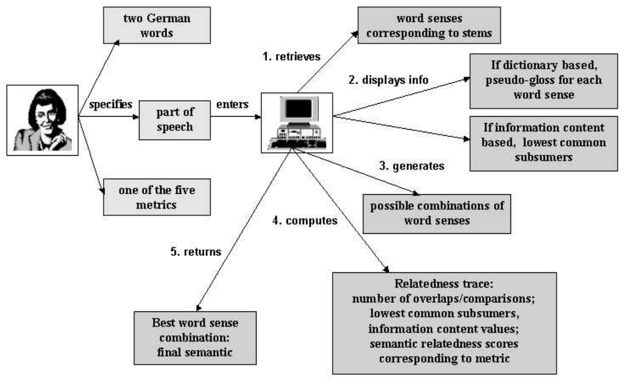

two German words
specifies
part of speech
enters
one of the five metrics
1. retrieves
word senses corresponding to stems
2. displays info
If dictionary based, pseudo-gloss for each word sense
If information content based, lowest common subsumers
3. generates
possible combinations of word senses
4. computes
Relatedness trace: number of overlaps/comparisons; lowest common subsumers, information content values; semantic relatedness scores corresponding to metric
5. returns
Best word sense combination: final semantic

Figure 1: The concept of user-system interaction.

## 4 Graphical User Interface

We developed a graphical user interface to interactively experiment with the software for computing semantic relatedness. The system runs on a standard Linux or Windows machine. Upon initialization, we configured the system to load an information content map computed from the German *taz* corpus.[5] The information content values encoded therein are employed by the information content based metrics. For the Lesk based metrics, two best configurations for generating definitions of word senses are offered via the GUI: one including three hypernyms of a word sense, and the other one including all related synsets (two iterations) except hyponyms. The representation of synsets in a generated definition is constituted by one (the first) of their word senses.

The user of the GUI can enter two words together with their part-of-speech and specify one of the five metrics. Then, the system displays the corresponding word stems, possible word senses ac-

[5]www.taz.de

cording to GermaNet, definitions generated for these word senses and their information content values. Furthermore, possible combinations of word senses for the two words are created and returned together with various diagnostic information specific to each of the metrics. This may be e.g. word overlaps in definitions for the Lesk based metrics, or lowest common subsumers and their respective information content values, depending on what is appropriate. Finally, the best word sense combination for the two words is determined and this is compactly displayed together with a semantic relatedness score. The interface allows the user to add notes to the results by directly editing the data shown in the GUI and save the detailed analysis in a text file for off-line inspection. The process of user-system interaction is summarized in Figure 1.

## 5 Conclusions

We presented software implementing an API to GermaNet and a case study built with this API, a package to compute five semantic relatedness metrics. We revised the metrics and in some cases redesigned them for the German language and GermaNet, as the latter is different from WordNet in a number of respects. The set of software functions resulting from our work is implemented in a JAVA library and can be used to build NLP applications with GermaNet or integrate GermaNet based semantic relatedness metrics into NLP systems. Also, we provide a graphical user interface which allows to interactively experiment with the system and study the performance of different metrics.

## Acknowledgments

This work has been funded by the Klaus Tschira Foundation. We thank Michael Strube for his valuable comments concerning this work.

## References

Fellbaum, Christiane (Ed.) (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Mass.: MIT Press.

Gurevych, Iryna (2005). *Using the Structure of a Conceptual Network in Computing Semantic Relatedness*. Submitted.

Gurevych, Iryna & Hendrik Niederlich (2005). Computing semantic relatedness of GermaNet concepts. In Bernhard Fisseni, Hans-Christian Schmitz, Bernhard Schröder & Petra Wagner (Eds.), *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Proceedings of Workshop "Applications of GermaNet II" at GLDV'2005*, pp. 462–474. Peter Lang.

Gurevych, Iryna & Michael Strube (2004). Semantic similarity applied to spoken dialogue summarization. In *Proceedings of the 20th International Conference on Computational Linguistics*, Geneva, Switzerland, 23 – 27 August 2004, pp. 764–770.

Hirst, Graeme & Alexander Budanitsky (2005). Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111.

Hundsnurscher, F. & J. Splett (1982). *Semantik der Adjektive im Deutschen: Analyse der semantischen Relationen*. Westdeutscher Verlag.

Jiang, Jay J. & David W. Conrath (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th International Conference on Research in Computational Linguistics (ROCLING)*. Tapei, Taiwan.

Kunze, Claudia & Lothar Lemnitzer (2002). GermaNet - representation, visualization, application. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Las Palmas, Canary Islands, Spain, 29 - 31 May, pp. 1485–1491.

Lemnitzer, Lothar & Claudia Kunze (2002). Adapting GermaNet for the Web. In *Proceedings of the fi rst Global WordNet Conference, Central Institute of Indian Languages. Mysore, India*, pp. 174–181.

Lesk, Michael (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, Toronto, Ontario, Canada, June, pp. 24–26.

Lin, Dekang (1998). An information-theoretic defi nition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, San Francisco, Cal., pp. 296–304.

McCarthy, Diana, Rob Koeling, Julie Weeds & John Carroll (2004). Finding predominant senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, 21–26 July 2004, pp. 280 – 287.

Patwardhan, Siddharth, Satanjeev Banerjee & Ted Pedersen (2003). Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, Mexico, pp. 241–257.

Pedersen, Ted, Siddharth Patwardhan & Jason Michelizzi (2004). WordNet::Similarity – Measuring the relatedness of concepts. In *Demonstrations of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Boston, Mass., 2–7 May 2004, pp. 267–270.

Resnik, Phil (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artifi cial Intelligence*, Montréal, Canada, 20–25 August 1995, Vol. 1, pp. 448–453.

Schmid, Helmut (1997). Probabilistic part-of-speech tagging using decision trees. In Daniel Jones & Harold Somers (Eds.), *New Methods in Language Processing*, Studies in Computational Linguistics, pp. 154–164. London, UK: UCL Press.

Vossen, Piek (1999). *EuroWordNet: a mutlilingual database with lexical-semantic networks*. Dordrecht: Kluwer Academic Publishers.

# Efficient solving and exploration of scope ambiguities

**Alexander Koller** and **Stefan Thater**
Department of Computational Linguistics
Saarland University, Saarbrücken, Germany
{koller,stth}@coli.uni-sb.de

## Abstract

We present the currently most efficient solver for scope underspecification; it also converts between different underspecification formalisms and counts readings. Our tool makes the practical use of large-scale grammars with (underspecified) semantic output more feasible, and can be used in grammar debugging.

## 1 Introduction

One of the most exciting recent developments in computational linguistics is that large-scale grammars which compute semantic representations are becoming available. Examples for such grammars are the HPSG English Resource Grammar (ERG) (Copestake and Flickinger, 2000) and the LFG ParGram grammars (Butt et al., 2002); a similar resource is being developed for the XTAG grammar (Kallmeyer and Romero, 2004).

But with the advent of such grammars, a phenomenon that is sometimes considered a somewhat artificial toy problem of theoretical semanticists becomes a very practical challenge: the presence of scope ambiguities. Because grammars often uniformly treat noun phrases as quantifiers, even harmless-looking sentences can have surprisingly many readings. The median number of scope readings for the sentences in the Rondane Treebank (distributed with the ERG) is 55, but the treebank also contains extreme cases such as (1) below, which according to the ERG has about 2.4 trillion ($10^{12}$) readings:

(1) Myrdal is the mountain terminus of the Flåm rail line (or Flåmsbana) which makes its way down the lovely Flåm Valley (Flåmsdalen) to its sea-level terminus at Flåm. (Rondane 650)

In order to control such an explosion of readings (and also to simplify the grammar design process), the developers of large-scale grammars typically use methods of packing or *underspecification* to specify the syntax-semantics interface. The general idea is that the parser doesn't compute all the individual scope readings, but only a compact *underspecified description*, from which the individual readings can then be extracted at a later stage of processing – but the underspecified description could also be used as a platform for the integration of lexical and context information, so as to restrict the set of possible readings without enumerating the wrong ones.

Such an approach is only feasible if we have access to efficient tools that support the most important operations on underspecified descriptions. We present `utool`, the Swiss Army Knife of Underspecification, which sets out to do exactly this. It supports the following operations:

1. enumerate all scope readings represented by an underspecified description;
2. check whether a description has any readings, and compute how many readings it has without explicitly enumerating them;
3. convert underspecified descriptions between different underspecification formalisms (at this point, Minimal Recursion Semantics (Copestake et al., 2003), Hole Semantics (Bos, 1996), and dominance constraints/graphs (Egg et al., 2001; Althaus et al., 2003)).

Our system is the fastest solver for underspecificied description available today; that is, it is fastest at solving Task 1 above (about 100.000 readings per second on a modern PC). It achieves this by implementing an efficient algorithm for solving dominance graphs (Bodirsky et al., 2004) and caching intermediate results in a chart data structure. To our knowledge, it is the *only* system that can do Tasks 2 and 3. It is only because `utool` can compute the number of readings without enumerating them that we even know that (1) has trillions of readings; even `utool` would take about a year to enumerate and count the readings individually.

`utool` is implemented in C++, efficient and portable, open source, and freely downloadable from `http://utool.sourceforge.net`.

## 2 Technical Description

### 2.1 Solving dominance graphs

At the core of `utool` is a solver for *dominance graphs* (Bodirsky et al., 2004) – graph representations of *weakly normal dominance constraints*, which constitute one of the main formalisms used in scope underspecification (Egg et al., 2001; Althaus et al., 2003). Dominance graphs are directed graphs with two kinds of edges, *tree edges* and *dominance edges*. They can be used to describe the set of all trees into which their tree edges can be embedded, in such a way that every dominance edge in the graph is realised as reachability in the tree. Dominance graphs are used as underspecified descriptions by describing sets of trees that are encodings of the formulas of some language of semantic representations, such as predicate logic.

Fig. 1 shows an example of a constraint graph for the sentence "every student reads a book." It consists of five *tree fragments* – sets of nodes that are connected by (solid) tree edges – which are connected by dominance edges (dotted lines). Two of the fragments have two *holes* each, into which other fragments can be "plugged". The graph can be embedded into the two trees shown in the middle of Fig. 1, which correspond to the two readings of the sentence. By contrast, the graph cannot be embedded into the tree shown on the right: a dominance edge stipulates that "read$_{x,y}$" must be reachable from "some$_y$", but it is not reachable from "some$_y$" in the tree. We call the two trees into which the graph can be embedded its *solutions*.

The Bodirsky et al. algorithm enumerates the solutions of a dominance graph (technically, its *solved forms*) by computing the set of its *free fragments*, which are the fragments that can occur at the root of some solution. Then it picks one of these fragments as the root and removes it from the graph. This splits the graph into several connected subgraphs, which are then solved recursively.

This algorithm can call itself for the same subgraph several times, which can waste a lot of time because the set of all solutions was already computed for the subgraph on the first recursive call. For this reason, our implementation caches intermediate results in a chart-like data structure. This data structure maps each subgraph $G$ to a set of *splits*, each of which records which fragment of $G$ should be placed at the root of the solution, what the subgraphs after removal of this fragment are, and how their solutions should be plugged into the holes of the fragment. In the worst case, the chart can have exponential size; but in practice, it is much smaller than the set of all solutions. For example, the chart for (1) contains 74.960 splits, which is a tiny number compared to the 2.4 trillion readings, and can be computed in a few seconds.

Now solving becomes a two-phase process. In the first phase, the chart data structure is filled by a run of the algorithm. In the second phase, the complete solutions are extracted from the chart. Although the first phase is conceptually much more complex than the second one because it involves interesting graph algorithms whose correctness isn't trivial to prove, it takes only a small fraction of the entire runtime in practice.

Instead of enumerating all readings from the chart, we can also compute the number of solutions represented by the chart. For each split, we compute the numbers of solutions of the fragment sets in the split. Then we multiply these numbers (choices for the children can be combined freely). Finally, we obtain the number of solutions for a subgraph by adding the numbers of solutions of all its splits. This computation takes linear time in the size of the chart.

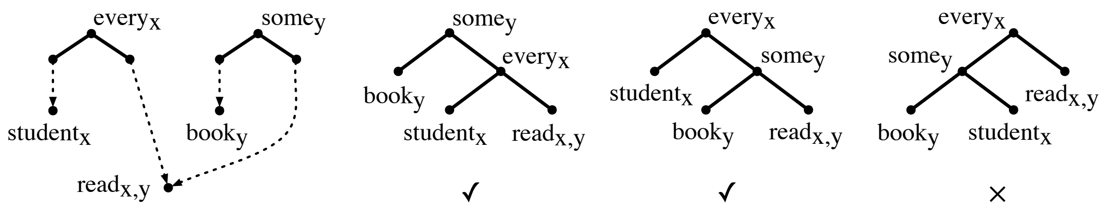


Figure 1: A dominance graph (left), two solutions (middle) and a non-solution (right).

## 2.2 Translating between formalisms

One of the most significant obstacles in the development of tools and resources for scope underspecification is that different resources (such as grammars and solvers) are built for different underspecification formalisms. To help alleviate this problem, `utool` can read and write underspecified descriptions and write out solutions in a variety of different formats:

- *dominance graphs*;
- descriptions of *Minimal Recursion Semantics*;
- descriptions of *Hole Semantics*.

The input and output functionality is provided by *codecs*, which translate between descriptions in one of these formalisms and the internal dominance graph format. The codecs for MRS and Hole Semantics are based on the (non-trivial) translations in (Koller et al., 2003; Niehren and Thater, 2003) and are only defined on *nets*, i.e. constraints whose graphs satisfy certain structural restrictions. This is not a very limiting restriction in practice (Flickinger et al., 2005). `utool` also allows the user to test efficiently whether a description is a net.

In practice, `utool` can be used to convert descriptions between the three underspecification formalisms. Because the codecs work with concrete syntaxes that are used in existing systems, `utool` can be used as a drop-in replacement e.g. in the LKB grammar development system (Copestake and Flickinger, 2000).

## 2.3 Runtime comparison

To illustrate `utool`'s performance, we compare its runtimes for the enumeration task with the (already quite efficient) MRS constraint solver of the LKB system (Copestake and Flickinger, 2000). Our data set consists of the 850 MRS-nets extracted from the Rondane treebank which have less than one million solutions (see Fig. 2). Fig. 3 displays the runtimes for enumerating all solutions, divided by the number of solutions, for both solvers. The horizontal axis shows the description sizes (number of tree fragments), and the (logarithmic!) vertical axis shows the average runtime per solution for descriptions of this size.

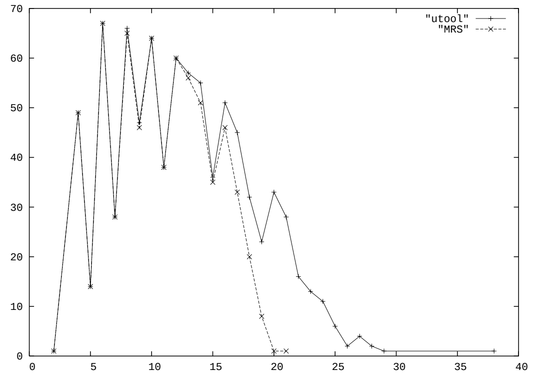


Figure 2: Distribution of constraints in Rondane over different sizes. The solid line shows the constraints in the data set, and the dashed line shows the constraints that the LKB solver could solve.

Due to memory limitations, the LKB solver could only solve descriptions with up to 21 tree fragments, which account for 80% of the test data. `utool` solved all descriptions in the test set. The evaluation was done using a 1.2 GHz PC with 2 GB of memory.

The figure shows that `utool` is generally faster than the LKB solver, up to a factor of approx. 1000. We should note that the LKB solver displays a dramatically higher variation in runtimes for constraints of the same size. Note that for small constraints, the runtimes tend to be too small to measure them accurately.

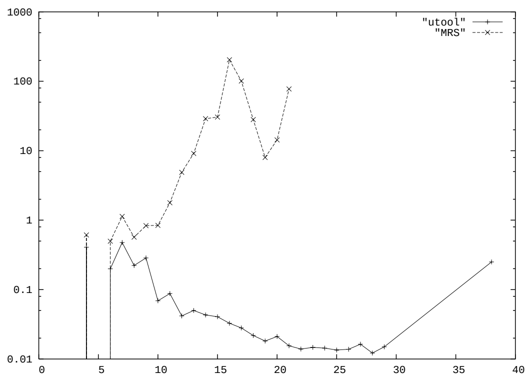


Figure 3: Runtimes per solution (in ms) for the MRS nets in the Rondane treebank for LKB and `utool`.

## 3 Conclusion

We have presented `utool`, a tool that supports a variety of operations related to scope underspecification. It is the most efficient solver for underspecification available today, and provides functionality for counting readings, testing whether a description is a net, and converting between different underspecification formalisms. It collects the results of several years of formal and computational research on dominance graphs into one convenient system.

The most obvious use of `utool` is the enumeration of readings of underspecified descriptions produced by large-scale grammars. This means that a user can realistically map the semantic output of these grammars into actual semantic representations. However, the tool is also useful for *developers* of such grammars. It can be used to count and explore the readings of the underspecified descriptions the grammar computes, and has already been used in the debugging of the syntax-semantics interface of the ERG (Flickinger et al., 2005).

From a more general perspective, the real appeal of underspecification is that it could allow us to eliminate readings that contradict the context or our world knowledge, without having to enumerate these readings first. Such inferences could already take place on the level of the underspecified description (Koller and Niehren, 2000). But the new chart data structure that `utool` computes is a more explicit packed representation of the possible readings, and still relatively small in practice. Thus it could open up avenues for more theoretical future research as well.

## References

Ernst Althaus, Denys Duchier, Alexander Koller, Kurt Mehlhorn, Joachim Niehren, and Sven Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48:194–219.

Manuel Bodirsky, Denys Duchier, Joachim Niehren, and Sebastian Miele. 2004. An efficient algorithm for weakly normal dominance constraints. In *ACM-SIAM Symposium on Discrete Algorithms*. The ACM Press.

Johan Bos. 1996. Predicate logic unplugged. In *Proceedings of the Tenth Amsterdam Colloquium*, pages 133–143.

Miriam Butt, Helge Dyvik, Tracey Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of the COLING 2002 Workshop on Grammar engeneering and evaluation*.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Conference on Language Resources and Evaluation*.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan Sag. 2003. Minimal recursion semantics: An introduction. Available at `http://lingo.stanford.edu/sag/papers/copestake.pdf`.

Markus Egg, Alexander Koller, and Joachim Niehren. 2001. The Constraint Language for Lambda Structures. *Logic, Language, and Information*, 10:457–485.

Dan Flickinger, Alexander Koller, and Stefan Thater. 2005. A new well-formedness criterion for semantics debugging. In *Proceedings of the 12th HPSG Conference*, Lisbon.

Laura Kallmeyer and Maribel Romero. 2004. LTAG semantics with semantic unification. In *Proceedings of the TAG+7 Workshop*, Vancouver.

Alexander Koller and Joachim Niehren. 2000. On underspecified processing of dynamic semantics. In *Proceedings of COLING-2000*, Saarbrücken.

Alexander Koller, Joachim Niehren, and Stefan Thater. 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *Proceedings of the 10th EACL*, Budapest.

Joachim Niehren and Stefan Thater. 2003. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

# CL Research's Knowledge Management System

Kenneth C. Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872
ken@clres.com
http://www.clres.com

## Abstract

CL Research began experimenting with massive XML tagging of texts to answer questions in TREC 2002. In DUC 2003, the experiments were extended into text summarization. Based on these experiments, The Knowledge Management System (KMS) was developed to combine these two capabilities and to serve as a unified basis for other types of document exploration. KMS has been extended to include web question answering, both general and topic-based summarization, information extraction, and document exploration. The document exploration functionality includes identification of semantically similar concepts and dynamic ontology creation. As development of KMS has continued, user modeling has become a key research issue: how will different users want to use the information they identify.

## 1 Introduction

In participating the TREC question-answering track, CL Research began by parsing full documents and developing databases consisting of semantic relation triples (Litkowski, 1999). The database approach proved to be quite confining, with time requirements expanding exponentially trying to maintain larger sets of documents and increasingly complex procedures to answer questions. A suggestion was made to tag text with the type of questions they could answer (e.g., tagging time phrases as answering **when** questions and person names as answering **who** questions). This led to the general approach of analyzing parse trees to construct an XML representation of texts (i.e., attaching metadata to the text) and examining these representations with XPath expressions to answer questions.

Litkowski (2003a) demonstrated the viability of this approach by showing that XPath expressions could be used to answer questions at a level above the highest performing team. Many issues and problems were identified: (1) The necessary level of analysis to meet the needs of particular applications; (2) tagging alternatives; and (3) the viability of the using the XML representation for text summarization, information extraction, novelty detection, and text mining. Subsequent efforts showed that XML representations could be effectively used in summarization (Litkowski, 2003b) and novelty detection (Litkowski, 2005).

Initially, CL Research developed an interface for examining question-answering performance. This interface has since evolved into a Knowledge Management System (KMS) that provides a single platform for examining English documents (e.g., newswire and research papers) and for generating different types of output (e.g., answers to questions, summaries, and document ontologies), also in XML representations. In this demonstration, CL Research will describe many parts of KMS, particularly the approaches used for analyzing texts.[1] The demonstration will particularly focus on the value of XML in providing a flexible and extensible mechanism for implementing the various NLP functionalities. In addition, the demonstration will identify the emerging issue of user modeling to determine exactly how knowledge will be used, since

[1]Screen shots of KMS in performing the functions as described below are can be seen at http://www.clres.com/kmsscreen.html.

the primary purpose of KMS is to serve as a tool that will enable users (such as scientists and intelligence analysts) to accumulate and manage knowledge (including facts, such as described in Fiszman et al., 2003) about topics of interest.[2]

## 2 Parsing and Creation of XML Tagging

KMS and each of its application areas is based on parsing text and then transforming parse trees into an XML representation. CL Research uses the Proximity Parser, developed by an inventor of top-down syntax-directed parsing (Irons, 1961).[3] The parser output consists of bracketed parse trees, with leaf nodes describing the part of speech and lexical entry for each sentence word. Annotations, such as number and tense information, may be included at any node. (Litkowski (2002) and references therein provide more details on the parser.)

After each sentence is parsed, its parse tree is traversed in a depth-first recursive function. During this traversal, each non-terminal and terminal node is analyzed to identify discourse segments (sentences and clauses), noun phrases, verbs, adjectives, and prepositional phrases. These items are maintained in lists; the growing lists constitute a document's discourse structure and are used, e.g., in resolving anaphora and establishing coreferents (implementing techniques inspired by Marcu (2000) and Tetreault (2001)). As these items are identified, they are subjected to a considerable amount of analysis to characterize them syntactically and semantically. The analysis includes word-sense disambiguation of nouns, verbs (including subcategorization identification), and adjectives and semantic analysis of prepositions to establish their semantic roles (such as described in Gildea & Jurafsky, 2002).

When all sentences of a document have been parsed and components identified and analyzed, the various lists are used to generate the XML representation. Most of the properties of the components are used as the basis for establishing XML attributes and values in the final representation. (Litkowski 2003a provides further details on this process.) This representation then becomes the basis for question answering, summarization, information extraction, and document exploration.

The utility of the XML representation does not stem from an ability to use XML manipulation technologies, such as XSLT and XQuery. In fact, these technologies seem to involve too much overhead. Instead, the utility arises within a Windows-based C++ development environment with a set of XML functions that facilitate working with node sets from a document's XML tree.

## 3 Question Answering

As indicated above, the initial implementation of the question-answering component of KMS was designed primarily to determine if suitable XPath expressions could be created for answering questions. CL Research's XML Analyzer was developed for this purpose.[4] XML Analyzer is constructed in a C++ Windows development environment to which a capability for examining XML nodes has been added. With this capability, a document can be loaded with one instruction and an XPath expression can be applied against this document in one more instruction to obtain a set of nodes which can be examined in more detail. Crucially, this enables low-level control over subsequent analysis steps (e.g., examining the text of a node with Perl regular expressions).

XML Analyzer first loads an XML file (which can include many documents, such as the "top 50" used in TREC). The user then presents an XPath expression and discourse components (typically, noun phrases) satisfying that expression are returned. XML Analyzer includes the document number, the sentence number, and the full sentence for each noun phrase. Several other features were added to XML Analyzer to examine characteristics of the documents and sentences (particularly to identify why an answer

[2]The overall design of KMS is based on requirements enunciated by intelligence analysts and question-answering researchers in a workshop on Scenario-Based Question Answering sponsored by the Advanced Research and Development Agency in 2003.

[3]An online demonstration of the parser is available at http://www.zzcad.com/parse.htm. A demo version of the parser is available for download at http://www.clres.com/demos.html.

[4]A demo version of XML Analyzer is available for download at http://www.clres.com/demos.html.

wasn't retrieved by an XPath expression).

XML Analyzer does not include the automatic creation of an XPath expression. KMS was created for TREC 2003 as the initial implementation of a complete question-answering system. In KMS, the question itself is parsed and transformed into an XML representation (using the same underlying functionality for processing documents) and then used to construct an XPath expression.

An XPath expression consists of two parts. The first part is a "passage retrieval" component, designed to retrieve sentences likely to contain the answer. This basic XPath is then extended for each question type with additional specifications, e.g., to ask for noun phrases that have time, location, or other semantic attributes. Experiments have shown that there is a tradeoff involved in these specifications. If they are very exacting, few possible answers are returned. Backoff strategies are used to return a larger set of potential answers and to analyze the context of these potential answers in more detail. The development of routines for automatic creation of XPath expressions is an ongoing process, but has begun to yield more consistent results (Litkowski, 2005).

In preparation for TREC 2004, KMS was further extended to incorporate a web-based component. With a check box to indicate whether the web or a document repository should be used, additional functionality was used to pose questions to Google. In web mode, an XML representation of a question is still developed, but then it is analyzed to present an optimal query to Google, typically, a pattern that will provide an answer. This involves the use of an integrated dictionary, particularly for creating appropriate inflected forms in the search query. KMS only uses the first page of Google results, without going into the source documents, extracting sentences from the Google results and using these as the documents. (A user can create a new "document repository" consisting of the documents from which answers have been obtained.) Many additional possibilities have emerged from initial explorations in using web-based question answering.

## 4 Summarization

Litkowski (2003a) indicated the possibility that the XML representation of documents could be used for summarization. To investigate this possibility, XML Analyzer was extended to include summarization capabilities for both general and topic-based summaries, including headline and keyword generation. Summarization techniques crucially take into account anaphora, coreferent, and definite noun phrase resolutions. As intimated in the analysis of the parse output, the XML representation for a referring expression is tagged with antecedent information, including both an identifying number and the full text of the antecedent. As a result, in examining a sentence, it is possible to consider the import of all its antecedents, instead of simply the surface form.

At the present time, only extractive summarization is performed in KMS. The basis for identifying important sentences is simply a frequency count of its words, but using antecedents instead of referring expressions. Stopwords and some other items are eliminated from this count.

In KMS, the user has the option for creating several kinds of summaries. The user specifies the type of summary (general, topic-based, headline, or keyword), which documents to summarize (one or many), and the length. Topic-based summaries require the user to enter search terms. The search terms can be as simple as a person's name or a few keywords or can be several sentences in length. Topic-based summaries use the search terms to give extra weight to sentences containing the search terms. Sentences are also evaluated for their novelty, with redundancy and overlap measures based on examining their noun phrases. KMS summarization procedures are described in more detail in Litkowski (2003b); novelty techniques are described in Litkowski (2005).

In KMS, summaries are saved in XML files as sets of sentences, each characterized by its source and sentence number. Each summary uses XML attributes containing the user's specifications and the documents included in the search. generated quickly but in whole form.

## 5 Document Exploration

KMS includes two major components for exploring the contents of a document. The first is based on the semantic types attached to nouns and verbs. The second is based on analyzing noun phrases to construct a document hierarchy or ontology.

As noted above, each noun phrase and each verb is tagged with its semantic class, based on WordNet. A user can explore one or more documents in three stages. First, a semantic category is specified. Second, the user pushes a button to obtain all the instances in the documents in that category. The phraseology in the documents is examined so that similar words (e.g., plurals and singulars and/or synonyms) are grouped together and then presented in a drop-down box by frequency. Finally, the user can select any term set and obtain all the sentences in the documents containing any of the terms.

KMS provides the capability for viewing a "dynamic" noun ontology of a document set. All noun phrases are analyzed into groups in a tree structure that portrays the ontology that is instantiated by these phrases. Noun phrases are reduced to their base forms (in cases of plurals) and grouped together first on the basis of their heads. Synonym sets are then generated and a further grouping is made. Algorithms from Navigli & Velardi (2004) are being modified and implemented in KMS. The user can then select a node in the ontology hierarchy and create a summary based on sentences containing any of its terms or children.

## 6 Dictionaries and Thesauruses in KMS

KMS makes extensive use of integrated dictionaries and thesauruses, in addition to a comprehensive dictionary used in parsing (which makes use of about 30 subcategorization patterns for verbs). This dictionary is supplemented with other dictionaries that are first used in dynamically extending the parser's dictionary for parsing, but then more extensively in semantic analysis. WordNet is used for many functions, as is a Roget-style thesaurus. KMS also uses a full machine-readable dictionary, dictionaries and semantic networks from the Unified Medical Language System, and a specially constructed dictionary of prepositions for semantic role analysis.

## 7 Summary

The preceding sections have focused on particular prominent functionalities (question-answering, summarization, and document exploration) in KMS. Each of these components is part of the whole, in which the main objective is to allow a user to explore documents in a variety of ways to identify salient portions of one or more documents. KMS is designed to identify relevant documents, to build a repository of these documents, to explore the documents, and to extract relevant pieces of information.

## References

Fiszman, M., Rindflesch, T., & Kilicoglu, H. (2003). Integrating a Hypernymic Proposition Interpreter into a Semantic Processor for Biomedical Texts. Proceedings of the AMIA Annual Symposium on Medical Informatics.

Gildea, Daniel, and Daniel Jurafsky. (2002) Automatic Labeling of Semantic Roles. *Computational Linguistics, 28* (3), 245-288.

Irons, E. T. (1961) A Syntax Directed Compiler for ALGOL-60. *Communications of the ACM, 4*, 51-55.

Litkowski, K. C. (1999). Question Answering Using Semantic Relation Triples. In E. M. Voorhees & D. K. Harman (eds.), *The Eighth Text Retrieval Conference (TREC-8)*. NIST Special Publication 500-246. Gaithersburg, MD., 349-56.

Litkowski, K. C. (2002). CL Research Experiments in TREC-10 Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 122-131.

Litkowski, K. C. (2003a). Question Answering Using XML-Tagged Documents. In E. M. Voorhees & L. P. Buckland (eds.), *The Eleventh Text Retrieval Conference (TREC 2002)*. NIST Special Publication 500-251. Gaithersburg, MD., 122-131.

Litkowski, K. C. (2003b). Text Summarization Using XML-Tagged Documents. Available: http://nlpir.nist.gov/projects/duc/pubs.html.

Litkowski, K. C. (2005). Evolving XML and Dictionary Strategies for Question Answering and Novelty Tasks. Available: http://trec.nist.gov/pubs/trec13/t13_proceedings.html.

Marcu, Daniel. (2000) The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics, 26* (3), 395-448.

Navigli, R. & P. Velardi (2004) Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. *Computational Linguistics 30*, 151-180.

Tetreault, Joel. (2001) A Corpus-Based Evaluation of Centering and Pronoun Resolution. *Computational Linguistics, 27* (4), 507-520.

# Dynamically Generating a Protein Entity Dictionary Using Online Resources

**Hongfang Liu**
Department of Information Systems
University of Maryland, Baltimore County
Baltimore, MD 21250
hfliu@umbc.edu

**Zhangzhi Hu** **Cathy Wu**
Department of Biochemistry and Molecular Biology
Georgetown University Medical Center
3900 Reservoir Road, NW, Washington, DC 20057
{zh9,wuc}@georgetown.edu

Abstract: With the overwhelming amount of biological knowledge stored in free text, natural language processing (NLP) has received much attention recently to make the task of managing information recorded in free text more feasible. One requirement for most NLP systems is the ability to accurately recognize biological entity terms in free text and the ability to map these terms to corresponding records in databases. Such task is called biological named entity tagging. In this paper, we present a system that automatically constructs a protein entity dictionary, which contains gene or protein names associated with UniProt identifiers using online resources. The system can run periodically to always keep up-to-date with these online resources. Using online resources that were available on Dec. 25, 2004, we obtained 4,046,733 terms for 1,640,082 entities. The dictionary can be accessed from the following website: http://biocreative.ifsm.umbc.edu/biothesaurus/.
Contact: hfliu@umbc.edu

## 1 Introduction

With the use of computers in storing the explosive amount of biological information, natural language processing (NLP) approaches have been explored to make the task of managing information recorded in free text more feasible *[1, 2]*. One requirement for NLP is the ability to accurately recognize terms that represent biological entities in free text. Another requirement is the ability to associate these terms with corresponding biological entities (i.e., records in biological databases) in order to be used by other automated systems for literature mining. Such task is called biological entity tagging. Biological entity tagging is not a trivial task because of several characteristics associated with biological entity names, namely: synonymy (i.e., different terms refer to the same entity), ambiguity (i.e., one term is associated with different entities), and coverage (i.e., entity terms or entities are not present in databases or knowledge bases).

Methods for biological entity tagging can be categorized into two types: one is to use a dictionary and a mapping method *[3-5]*, and the other is to markup terms in the text according to contextual cues, specific verbs, or machine learning *[6-10]*. The performance of biological entity tagging systems using dictionaries depends on the coverage of the dictionary as well as mapping methods that can handle synonymous or ambiguous terms. Strictly speaking, tagging systems that do not use dictionaries are not biological entity tagging but biological term tagging, since tagged terms in text are not associated with specific biological entities stored in databases. It requires an additional step to map terms mentioned in the text to records in biological databases in order to be automatically integrated with other system or databases. Due to the dynamic nature associated with the molecular biology domain, it is critical to have a comprehensive biological entity dictionary that is always up-to-date.

In this paper, we present a system that constructs a large protein entity dictionary, BioThesaurus, using online resources. Terms in the dictionary are then curated based on high ambiguous terms to flag nonsensical terms (e.g., *Novel protein*) and are also curated based on the semantic categories acquired from the UMLS to flag descriptive terms that associate with other semantic types other than gene or proteins (e.g., terms that refer to species, cells or other small molecules). In the following, we first provide background and related work on dictionary construction using online resources. We then present our method on constructing the dictionary.

## 2 Resources

The system utilizes several large size biological databases including three NCBI databases (GenPept [11], RefSeq [12], and Entrez GENE [13]), PSD database from Protein Information Resources (PIR) [14], and

UniProt [15]. Additionally, several model organism databases or nomenclature databases were used. Correspondences among records from these databases are identified using the rich cross-reference information provided by the iProClass database of PIR [14]. The following provides a brief description of each of the database.

**PIR Resources** – There are three databases in PIR: the Protein Sequence Database (PSD), iProClass, and PIR-NREF. PSD database includes functionally annotated protein sequences. The iProClass database is a central point for exploration of protein information, which provides summary descriptions of protein family, function and structure for all protein sequences from PIR, Swiss-Prot, and TrEMBL (now UniProt). Additionally, it links to over 70 biological databases in the world. The PIR-NREF database is a comprehensive database for sequence searching and protein identification. It contains non-redundant protein sequences from PSD, Swiss-Prot, TrEMBL, RefSeq, GenPept, and PDB.

**UniProt** – UniProt provides a central repository of protein sequence and annotation created by joining Swiss-Prot, TrEMBL, and PSD. There are three knowledge components in UniProt: Swissprot, TrEMBL, and UniRef. Swissprot contains manually-annotated records with information extracted from literature and curator-evaluated computational analysis. TrEMBL consists of computationally analyzed records that await full manual annotation. The UniProt Non-redundant Reference (UniRef) databases combine closely related sequences into a single record where similar sequences are grouped together. Three UniRef tables UniRef100, UniRef90 and UniRef50) are available for download: UniRef100 combines identical sequences and sub-fragments into a single UniRef entry; and UniRef90 and UniRef50 are built by clustering UniRef100 sequences into clusters based on the CD-HIT algorithm [16] such that each cluster is composed of sequences that have at least 90% or 50% sequence similarity, respectively, to the representative sequence.

**NCBI resources** – three data sources from NCBI were used in this study: GenPept, RefSeq, and Entrez GENE. GenPept entries are those translated from the GenBanknucleotide sequence database. RefSeq is a comprehensive, integrated, non-redundant set of sequences, including genomic DNA, transcript (RNA), and protein products, for major research organisms. Entrez GENE provides a unified query environment for genes defined by sequence and/or in NCBI's Map Viewer. It records gene names, symbols, and many other attributes associated with genes and the products they encode.

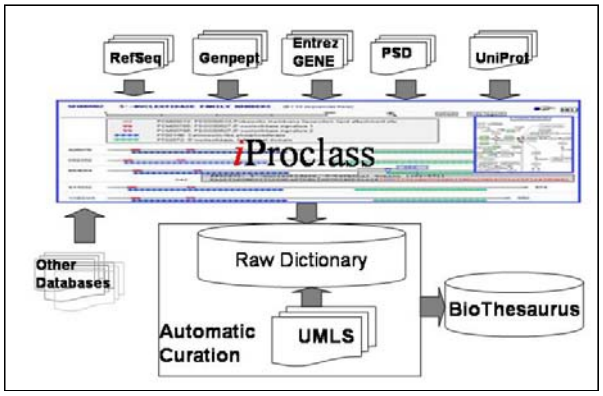


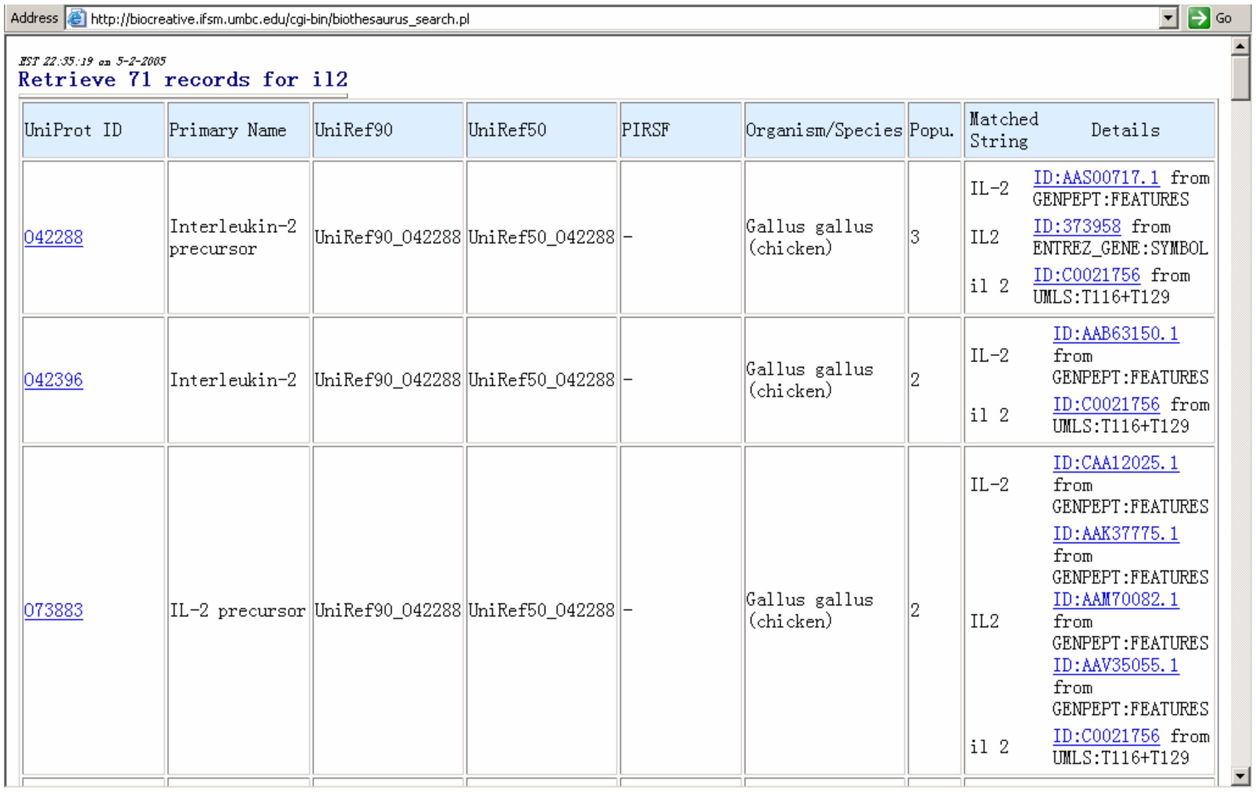
Figure 1: The overall architecture of the system

**The UMLS** – the Unified Medical Language System (UMLS) has been developed and maintained by National Library of Medicine (NLM) [17]. It contains three knowledge sources: the Metathesaurus (META), the SPECIALIST lexicon, and the Semantic Network. The META provides a uniform, integrated platform for over 60 biomedical vocabularies and classifications, and group different names for the same concept. The SPECIALIST lexicon contains syntactic information for many terms, component words, and English words, including verbs, which do not appear in the META. The Semantic Network contains information about the types or categories (e.g., “Disease or Syndrome”, “Virus”) to which all META concepts have been assigned.

**Other molecular biology databases** - We also included several model organism databases or nomenclature databases in the construction of the dictionary, i.e., mouse - Mouse Genome Database (MGD) [18], fly - FlyBase [19], yeast - Saccharomyces Genome Database (SGD) [20], rat – Rat Genome Database (RGD) [21], worm – WormBase [22], Human Nomenclature Database (HUGO) [23], Online Mendelian Inheritance in Man (OMIM) [24], and Enzyme Nomenclature Database (ECNUM) [25, 26].

## 3 System Description and Results

The system was developed using PERL and the PERL module Net::FTP. Figure 1 depicts the overall architecture. It automatically gathers fields that contain annotation information from PSD, RefSeq, Swiss-Prot, TrEMBL, GenBank, Entrez GENE, MGI, RGD, HUGO, ENCUM, FlyBase, and WormBase for each iProClass record from the distribution website


Address
http://biocreative.ifsm.umbc.edu/cgi-bin/biothesaurus_search.pl
Go

EST 22:35:19 on 5-2-2005

Retrieve 71 records for il2

| UniProt ID | Primary Name | UniRef90 | UniRef50 | PIRSF | Organism/Species | Popu. | Matched String | Details |
|---|---|---|---|---|---|---|---|---|
| O42288 | Interleukin-2 precursor | UniRef90_O42288 | UniRef50_O42288 | - | Gallus gallus (chicken) | 3 | IL-2<br>IL2<br>il 2 | ID:AAS00717.1 from GENPEPT:FEATURES<br>ID:373958 from ENTREZ_GENE:SYMBOL<br>ID:C0021756 from UMLS:T116+T129 |
| O42396 | Interleukin-2 | UniRef90_O42288 | UniRef50_O42288 | - | Gallus gallus (chicken) | 2 | IL-2<br>il 2 | ID:AAB63150.1 from GENPEPT:FEATURES<br>ID:C0021756 from UMLS:T116+T129 |
| O73883 | IL-2 precursor | UniRef90_O42288 | UniRef50_O42288 | - | Gallus gallus (chicken) | 2 | IL-2<br>IL2<br>il 2 | ID:CAA12025.1 from GENPEPT:FEATURES<br>ID:AAK37775.1 from GENPEPT:FEATURES<br>ID:AAM70082.1 from GENPEPT:FEATURES<br>ID:AAV35055.1 from GENPEPT:FEATURES<br>ID:C0021756 from UMLS:T116+T129 |

Figure 2: Screenshot of retrieving il2 from BioThesaurus

of each resource. Annotations extracted from each resource were then processed to extract terms where each term is associated with one or more UniProt unique identifiers and comprised the raw dictionary for BioThesaurus. The raw dictionary was computationally curated using the UMLS to flag the UMLS semantic types and remove several high frequent nonsensical terms. There were a total of 1,677,162 iProclass records in the PIR release 59 (released on Dec 25 2004). From it, we obtained 4,046,733 terms for 1,640,082 entities. Note that about 27,000 records have no terms in the dictionary mostly because they are new sequences and have not been annotated and linked to other resources or terms associated with them are nonsensical. The dictionary can be searched through the following URL: http://biocreative.ifsm.umbc.edu/biothesaurus/Biothesaurus.html.

Figure 2 shows a screenshot when retrieving entities associated with term il2. It indicates that there are totally 71 entities in UniProt that il2 represents when ignoring textual variants. The first column of the table is UniProt ID. The primary name is shown in the second column, the family classifications available from iProClass are shown in the following several columns, the taxonomy information is shown in the next. The popularity of the term (i.e., the number of databases that contain the term or its variants) is shown next. And the last column shows the links to the records from which the system extracted the terms.

## 4 Discussion and Conclusion

We demonstrated here a system which generates a protein entity dictionary dynamically using online resources. The dictionary can be used by biological entity tagging systems to map entity terms mentioned in the text to specific records in UniProt.

**Acknowledgements**

The project was supported by IIS-0430743 from the National Science Foundation.

## Reference

1. Hirschman L, Park JC, Tsujii J, Wong L, Wu CH: **Accomplishments and challenges in literature data mining for biology**. *Bioinformatics* 2002, **18**(12):1553-1561.

2. Shatkay H, Feldman R: **Mining the biomedical literature in the genomic era: an overview**. *J Comput Biol* 2003, **10**(6):821-855.
3. Krauthammer M, Rzhetsky A, Morozov P, Friedman C: **Using BLAST for identifying gene and protein names in journal articles**. *Gene* 2000, **259**(1-2):245-252.
4. Jenssen TK, Laegreid A, Komorowski J, Hovig E: **A literature network of human genes for high-throughput analysis of gene expression**. *Nat Genet* 2001, **28**(1):21-28.
5. Hanisch D, Fluck J, Mevissen HT, Zimmer R: **Playing biology's name game: identifying protein names in scientific text**. *Pac Symp Biocomput* 2003:403-414.
6. Fukuda K, Tamura A, Tsunoda T, Takagi T: **Toward information extraction: identifying protein names from biological papers**. *Pac Symp Biocomput* 1998:707-718.
7. Sekimizu T, Park HS, Tsujii J: **Identifying the Interaction between Genes and Gene Products Based on Frequently Seen Verbs in Medline Abstracts**. *Genome Inform Ser Workshop Genome Inform* 1998, **9**:62-71.
8. Narayanaswamy M, Ravikumar KE, Vijay-Shanker K: **A biological named entity recognizer**. *Pac Symp Biocomput* 2003:427-438.
9. Tanabe L, Wilbur WJ: **Tagging gene and protein names in biomedical text**. *Bioinformatics* 2002, **18**(8):1124-1132.
10. Lee KJ, Hwang YS, Kim S, Rim HC: **Biomedical named entity recognition using two-phase model based on SVMs**. *J Biomed Inform* 2004, **37**(6):436-447.
11. Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Wheeler DL: **GenBank: update**. *Nucleic Acids Res* 2004, **32 Database issue**:D23-26.
12. Pruitt KD, Katz KS, Sicotte H, Maglott DR: **Introducing RefSeq and LocusLink: curated human genome resources at the NCBI**. *Trends Genet* 2000, **16**(1):44-47.
13. NCBI: **Entrez Gene**. In., vol. http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene; 2004.
14. Wu CH, Yeh LS, Huang H, Arminski L, Castro-Alvear J, Chen Y, Hu Z, Kourtesis P, Ledley RS, Suzek BE *et al*: **The Protein Information Resource**. *Nucleic Acids Res* 2003, **31**(1):345-347.
15. Apweiler R, Bairoch A, Wu CH, Barker WC, Boeckmann B, Ferro S, Gasteiger E, Huang H, Lopez R, Magrane M *et al*: **UniProt: the Universal Protein knowledgebase**. *Nucleic Acids Res* 2004, **32 Database issue**:D115-119.
16. Li W, Jaroszewski L, Godzik A: **Clustering of highly homologous sequences to reduce the size of large protein databases**. *Bioinformatics* 2001, **17**(3):282-283.
17. Bodenreider O: **The Unified Medical Language System (UMLS): integrating biomedical terminology**. *Nucleic Acids Res* 2004, **32 Database issue**:D267-270.
18. Bult CJ, Blake JA, Richardson JE, Kadin JA, Eppig JT, Baldarelli RM, Barsanti K, Baya M, Beal JS, Boddy WJ *et al*: **The Mouse Genome Database (MGD): integrating biology with the genome**. *Nucleic Acids Res* 2004, **32 Database issue**:D476-481.
19. Consortium F: **The FlyBase database of the Drosophila genome projects and community literature**. *Nucleic Acids Res* 2003, **31**(1):172-175.
20. Cherry JM, Adler C, Ball C, Chervitz SA, Dwight SS, Hester ET, Jia Y, Juvik G, Roe T, Schroeder M *et al*: **SGD: Saccharomyces Genome Database**. *Nucleic Acids Res* 1998, **26**(1):73-79.
21. Twigger S, Lu J, Shimoyama M, Chen D, Pasko D, Long H, Ginster J, Chen CF, Nigam R, Kwitek A *et al*: **Rat Genome Database (RGD): mapping disease onto the genome**. *Nucleic Acids Res* 2002, **30**(1):125-128.
22. Harris TW, Chen N, Cunningham F, Tello-Ruiz M, Antoshechkin I, Bastiani C, Bieri T, Blasiar D, Bradnam K, Chan J *et al*: **WormBase: a multi-species resource for nematode biology and genomics**. *Nucleic Acids Res* 2004, **32 Database issue**:D411-417.
23. Povey S, Lovering R, Bruford E, Wright M, Lush M, Wain H: **The HUGO Gene Nomenclature Committee (HGNC)**. *Hum Genet* 2001, **109**(6):678-680.
24. Hamosh A, Scott AF, Amberger JS, Bocchini CA, McKusick VA: **Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders**. *Nucleic Acids Res* 2005, **33 Database Issue**:D514-517.
25. Gegenheimer P: **Enzyme nomenclature: functional or structural?** *Rna* 2000, **6**(12):1695-1697.
26. Tipton K, Boyce S: **History of the enzyme nomenclature system**. *Bioinformatics* 2000, **16**(1):34-40.

# Descriptive Question Answering in Encyclopedia

**Hyo-Jung Oh, Chung-Hee Lee, Hyeon-Jin Kim, Myung-Gil Jang**
Knowledge Mining Research Team
Electronics and Telecommunications Research Institute (ETRI)
Daejeon, Korea
{ohj, forever, jini, mgjang} @ etri.re.kr

## Abstract

Recently there is a need for a QA system to answer not only factoid questions but also descriptive questions. Descriptive questions are questions which need answers that contain definitional information about the search term or describe some special events. We have proposed a new descriptive QA model and presented the result of a system which we have built to answer descriptive questions. We defined 10 Descriptive Answer Type(DAT)s as answer types for descriptive questions. We discussed how our proposed model was applied to the descriptive question with some experiments.

## 1 Introduction

Much of effort in Question Answering has focused on the 'short answers' or factoid questions, which answer questions for which the correct response is a single word or short phrase from the answer sentence. However, there are many questions which are better answer with a longer description or explanation in logs of web search engines(Voorhees, 2003). In this paper, we introduce a new descriptive QA model and present the result of a system which we have built to answer such questions.

Descriptive question are questions such as "Who is Columbus?", "What is tsunami?", or "Why is blood red?", which need answer that contain the definitional information about the search term, explain some special phenomenon.(i.e. chemical reaction) or describe some particular events.

At the recent works, definitional QA, namely questions of the form "What is X?", is a developing research area related with a subclass of descriptive questions. Especially in TREC-12 conference(Voorhees, 2003), they had produced 50 definitional questions in QA track for the competition. The systems in TREC-12(Blair et al, 2003; Katz et al, 2004) applied complicated technique which was integrated manually constructed definition patterns with statistical ranking component.

Some experiments(Cui et al, 2004) tried to use external resources such as WordNet and Web Dictionary associated with a syntactic pattern. Further recent work tried to use online knowledge bases on web. Domain-specific definitional QA systems in the same context of our works have been developed. Shiffman et al(2001) applied on biographical summaries for people with data-driven method.

In contrast to former research, we focus on the other descriptive question, such as "why," "how," and "what kind of". We also present our descriptive QA model and its experimental results.

## 2 Descriptive QA

### 2.1 Descriptive Answer Type

Our QA system is a domain specific system for encyclopedia[1]. One of the characteristics of encyclopedia is that it has many descriptive sentences. Because encyclopedia contains facts about many different subjects or about one particular subject explained for reference, there are

[1] Our QA system can answer both factoid questions and descriptive questions. In this paper, we present only sub system for descriptive QA

many sentences which present definition such as "X is Y." On the other hand, some sentences describe process of some special event(i.e. the 1st World War) so that it forms particular sentence structures like news article which reveal reasons or motives of the event.

We defined Descriptive Answer Type (DAT) as answer types for descriptive questions with two points of view: *what kind of descriptive questions are in the use's frequently asked questions?* and *what kind of descriptive answers can be patternized in the our corpus?* On the view of question, most of user's frequently asked questions are not only factoid questions but also definitional questions. Furthermore, the result of analyzing the logs of our web site shows that there are many questions about 'why', "how', and so on. On the other side, descriptive answer sentences in corpus show particular syntactic patterns such as appositive clauses, parallel clauses, and adverb clauses of cause and effect. In this paper, we defined 10 types of DAT to reflect these features of sentences in encyclopedia.

Table 1 shows example sentences with pattern for each DAT. For instance, "A tsunami is a large wave, often caused by an earthquake." is an example for 'Definition' DAT with pattern of [X is Y]. It also can be an example for 'Reason' DAT because of matching pattern of [X is caused by Y].

Table 1: Descriptive Answer Type

| DAT | Example/Pattern |
|---|---|
| DEFINITION | *A tsunami* is a large wave, often caused by an earthquake. [X is Y] |
| FUCTION | *Air bladder* is an air-filled structure in many fishes that functions to maintain buoyancy or to aid in respiration. [ X that function to Y] |
| KIND | *The coins* in States are 1 cent, 5 cents, 25 cents, and 100cents. [X are $Y_1$, $Y_2$,.. and $Y_n$] |
| METHOD | The method that *prevents a cold* is washing often your hand.[The method that/of X is Y] |
| CHARCTER | *Sea horse*, characteristically swimming in an upright position and having a prehensile tail. [ X is characteristically Y] |
| OBJECTIVE | *An automobile* used for land transports. [ X used for Y] |
| REASON | *A tsunami* is a large wave, often caused by an earthquake. [X is caused by Y] |
| COMPONENT | *An automobile* usually is composed of 4 wheels, an engine, and a steering wheel. [X is composed of $Y_1$, $Y_2$,.. and $Y_n$] |
| PRINCIPLE | *Osmosis* is the principle, transfer of a liquid solvent through a semipermeable membrane that does not allow dissolved solids to pass. [X is the principle, Y] |
| ORIGIN | *The Achilles tendon* is the name from the mythical Greek hero Achilles. [X is the name from Y] |

## 2.2 Descriptive Answer Indexing

Descriptive Answer indexing process consists of two parts: *pattern extraction* from pre-tagged corpus and *extraction of DIU*(Descriptive Indexing Unix) using a pattern matching technique.

Descriptive answer sentences generally have a particular syntactic structure. For instance, definitional sentences has patterns such as "X is Y," "X is called Y," and "X means Y." In case of sentence which classifies something into sub-kinds, i.e. "Our coin are 50 won, 100 won and 500 won." it forms parallel structure like "X are $Y_1$, $Y_2$,.. and $Y_n$".

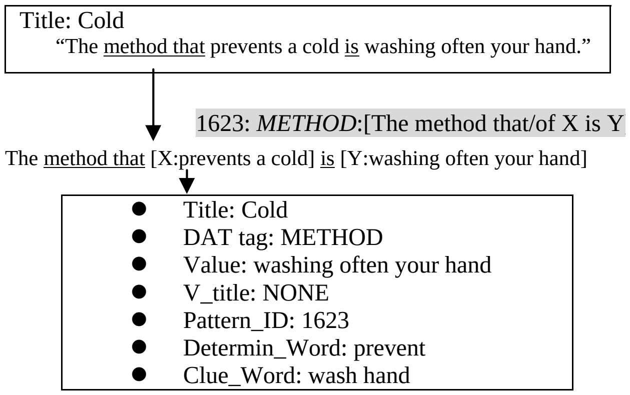
To extract these descriptive patterns, we first build initial patterns. We constructed pre-tagged corpus with 10 DAT tags, then performed sentence alignment by the surface tag boundary. The tagged sentences are then processed through part-of-speech(POS) tagging in the first step. In this stage, we can get descriptive clue terms and structures, such as "X is caused by Y" for 'Reason', 'X was made for Y" for 'Function', and so on.

In the second step, we used linguistic analysis including chunking and parsing to extend initial patterns automatically. Initial patterns are too rigid because we look up only surface of sentences in the first step. If some clue terms appear with long distance in a sentence, it can fail to be recognized as a pattern. To solve this problem, we added sentence structure patterns on each DAT patterns, such as appositive clause patterns for 'Definition', parallel clause patterns for 'Kind', and so on.

Finally, we generalized patterns to conduct flexible pattern matching. We need to group patterns to adapt to various variations of terms which appear in un-training sentences. Several similar patterns under the same DAT tag were integrated into regular-expression union which is to be formulated automata. For example, 'Definition' patterns are represented by [X<NP> be called/named/known as Y<NP>].

We defined DIU as indexing unit for descriptive answer candidate. In DIU indexing stage performed pattern matching, extracting DIU, and storing our storage. We built a pattern matching system based on Finite State Automata(FSA). After pattern matching, we need to filtering over-generated candidates because descriptive patterns are naive in a sense. In case of 'Definition', "X is Y" is matched so many times, that we restrict the pattern when "X" and "Y" under the same meaning on our ETRI-LCN for Noun ontology [2]. For example, "Customs duties are taxes that people pay for importing and exporting goods[X is Y]" are accepted because 'custom duty' is under the 'tax' node so they have same meaning.

DIU consists of Title, DAT tag, Value, V_title, Pattern_ID, Determin_word, and Clue_word. Title and Value means X and Y in result of pattern matching, respectively. Determin_word and Clue_word are used to restrict X and Y in the retrieval stage, respectively. V_title is distinguished from Title by whether X is an entry in the encyclopedia or not. Figure 1 illustrated result of extracting DIU.


Title: Cold
"The method that prevents a cold is washing often your hand."

1623: *METHOD*:[The method that/of X is Y

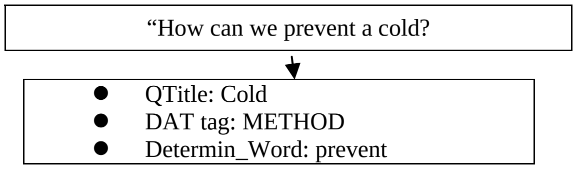
The method that [X:prevents a cold] is [Y:washing often your hand]

- Title: Cold
- DAT tag: METHOD
- Value: washing often your hand
- V_title: NONE
- Pattern_ID: 1623
- Determin_Word: prevent
- Clue_Word: wash hand

Figure 1: Result of DIU extracting

## 2.3 Descriptive Answer Retrieval

Descriptive answer retrieval performs finding DIU candidates which are appropriate to user questions through query processing. The important role of query processing is to catch out <QTitle, DAT> pair in the user question. QTitle means the key search word in a question. We used LSP pattern[3] for question analysis. Another function of query processing is to extract Determin_word or Clue_Terms in question in terms of determining what user questioned. Figure 2 illustrates the result of QDIU(Question DIU).


"How can we prevent a cold?

- QTitle: Cold
- DAT tag: METHOD
- Determin_Word: prevent

Figure 2: Result of Question Analysis

[2] LCN: Lexical Concept Network. ETRI-LCN for Noun consists of 120,000 nouns and 224,000 named entities.
[3] LSP pattern: Lexico-Syntactic Pattern. We built 774 LSP patterns.

## 3 Experiments

### 3.1 Evaluation of DIU Indexing

To extract descriptive patterns, we built 1,853 pre-tagged sentences within 2,000 entries. About 40%(760 sentences) of all are tagged with 'Definition, while only 9 sentences were assigned to 'Principle'. Table 2 shows the result of extracted descriptive patterns using tagged corpus. 408 patterns are generated for 'Definition' from 760 tagged sentences, while 938 patterns for 'Function' from 352 examples. That means the sentences of describing something's function formed very diverse expressions.

Table 2: Result of Descriptive Pattern Extraction

| DAT | # of Patterns | DAT | # of Patterns |
|---|---|---|---|
| DEFINITION | 408(22) | OBJECTIVE | 166(22) |
| FUCTION | 938(26) | REASON | 38(15) |
| KIND | 617(71) | COMPONENT | 122(19) |
| METHOD | 104(29) | PRINCIPLE | 3(3) |
| CHARCTER | 367(20) | ORIGIN | 491(52) |
| | | Total | 3,254(279) |

* The figure in ( ) means # of groups of patterns

Table 3: Result of DIU Indexing

| DAT | # of DIUs | DAT | # of DIUs |
|---|---|---|---|
| DEFINITION | 164,327(55%) | OBJECTIVE | 9,381(3%) |
| FUCTION | 25,105(8%) | REASON | 17,647(6%) |
| KIND | 45,801(15%) | COMPONENT | 12,123(4%) |
| METHOD | 4,903(2%) | PRINCIPLE | 64(0%) |
| CHARCTER | 10,397(3%) | ORIGIN | 10,504(3%) |
| | | Total | 300,252 |

Table 3 shows the result of DIU indexing. We extracted 300,252 DIUs from the whole encyclopedia [4] using our Descriptive Answer Indexing process. As expected, most DIUs(about 55%, 164,327 DIUs) are 'Definition'. We assumed that the entries belonging to the 'History' category have many sentences about 'Reason' because history usually describes some events. However, we obtained only 25,110 DIUs(8%) of 'Reason' because patterns of 'Reason' have lack of expressing syntactic structure of adverb clauses of cause and effect. 'Principle' also has same problem of lack of patterns so we only 64 DIUs.

### 3.2 Evaluation of DIU Retrieval

To evaluate our descriptive question answering method, we used 152 descriptive questions from our ETRI QA Test Set 2.0[5], judged by 4 assessors.

[4] Our encyclopedia consists of 163,535 entries and 13 main categories in Korean.
[5] ETRI QA Test Set 2.0 consists of 1,047 <question, answer> pairs including both factoid and descriptive questions for all categories in encyclopedia

For performance comparisons, we used Top 1 and Top 5 precision, recall and F-score. Top 5 precision is a measure to consider whether there is a correct answer in top 5 ranking or not. Top 1 measured only one best ranked answer.

For our experimental evaluations we constructed an operational system in the Web, named "AnyQuestion 2.0." To demonstrate how effectively our model works, we compared to a sentence retrieval system. Our sentence retrieval system used vector space model for query retrieval and 2-poisson model for keyword weighting.

Table 4 shows that the scores using our proposed method are higher than that of traditional sentence retrieval system. As expected, we obtained better result(0.608) than sentence retrieval system(0.508). We gain 79.3% (0.290 to 0.520) increase on Top1 than sentence retrieval and 19.6%(0.508 to 0.608) on Top5. The fact that the accuracy on Top1 has dramatically increased is remarkable, in that question answering wants exactly only one relevant answer.

Whereas even the recall of sentence retrieval system(0.507) is higher than descriptive QA result(0.500) on Top5, the F-score(0.508) is lower than that(0.608). It comes from the fact that sentence retrieval system tends to produce more number of candidates retrieved. While sentence retrieval system retrieved 151 candidates, our descriptive QA method retrieved 98 DIUs under the same condition that the number of corrected answers of sentence retrieval is 77 and ours is 76.

Table 4: Result of Descriptive QA

| | Sentence Retrieval | | Descriptive QA | |
|---|---|---|---|---|
| | Top l | Top 5 | Top 1 | Top 5 |
| Retrieved | 151 | 151 | 98 | 98 |
| Corrected | 44 | 77 | 65 | 76 |
| Precision | 0.291 | 0.510 | 0.663 | 0.776 |
| Recall | 0.289 | 0.507 | 0.428 | 0.500 |
| F-score | 0.290 | 0.508 | 0.520 (+79.3%) | 0.608 (+19.6%) |

We further realized that our system has a few week points. Our system is poor for inverted retrieval which should answer to the quiz style questions, such as "What is a large wave, often caused by an earthquake?" Moreover, our system depends on initial patterns. For the details, 'Principle' has few initial patterns, so that it has few descriptive patterns. This problem has influence on retrieval results, too.

## 4 Conclusion

We have proposed a new descriptive QA model and presented the result of a system which we have built to answer descriptive questions. To reflect characteristics of descriptive sentences in encyclopedia, we defined 10 types of DAT as answer types for descriptive questions. We explained how our system constructed descriptive patterns and how these patterns are worked on our indexing process. Finally we presented how descriptive answer retrieval performed and retrieved DIU candidates. We have shown that our proposed model outperformed the traditional sentence retrieval system with some experiments. We obtained F-score of 0.520 on Top1 and 0.680 on Top5. It showed better results when compared with sentence retrieval system on both Top1 and Top5.

Our Further works will concentrate on reducing human efforts for building descriptive patterns. To achieve automatic pattern generation, we will try to apply machine learning technique like the boosting algorithm. More urgently, we have to build an inverted retrieval method. Finally, we will compare with other systems which participated in TREC by translating definitional questions of TREC in Korean.

## References

S. Blair-Goldensohn, K. R. McKeown, and A, H, Schlaikjer. 2003. *A Hybrid Approach for QA Track Definitional Questions*, Proceedings of the twelve Text REtreival Conference(TREC-12), pp. 336-342.

H. Cui, M-Y. Kan, T-S. Chua, and J. Xian. 2004. *A Comparative Study on Sentence Retrieval for Definitional Question Answering*, Proceedings of SIGIR 2004 workshop on Information Retrieval 4 Question Answering(IR4QA).

B. Katz, M. Bilotti, S. Felshin, et. al. 2004. *Answering Multiple Questions on a Topic from Heterogeneous Resources*, Proceedings of the thirteenth Text REtreival Conference(TREC-13).

B. Shiffman, I. Mani, and K.Concepcion. 2001. *Producing Biographical Summaries: Combining Linguistic Resources and Corpus Statistics*, Proceedings of the European Association for Computational Linguistics (ACL-EACL 01).

Ellen M. Voorhees. 2003. *Overview of TREC 2003 Question Answering Track*, Proceedings of the twelfth Text REtreival Conference(TREC-12).

# High Throughput Modularized NLP System for Clinical Text

**Serguei Pakhomov**
Mayo College of Medicine

Mayo Clinic
Rochester, MN, 55905
pakhomov@mayo.edu

**James Buntrock**
Division of Biomedical Informatics
Mayo Clinic
Rochester, MN, 55905
Buntrock@mayo.edu

**Patrick Duffy**
Division of Biomedical Informatics
Mayo Clinic
Rochester, MN, 55905
duffp@mayo.edu

## Abstract

This paper presents the results of the development of a high throughput, real time modularized text analysis and information retrieval system that identifies clinically relevant entities in clinical notes, maps the entities to several standardized nomenclatures and makes them available for subsequent information retrieval and data mining. The performance of the system was validated on a small collection of 351 documents partitioned into 4 query topics and manually examined by 3 physicians and 3 nurse abstractors for relevance to the query topics. We find that simple key phrase searching results in 73% recall and 77% precision. A combination of NLP approaches to indexing improve the recall to 92%, while lowering the precision to 67%.

## 1 Introduction

Until recently the NLP systems developed for processing clinical texts have been narrowly focused on a specific type of document such as radiology reports [1], discharge summaries [2], medline abstracts [3], pathology reports [4]. In addition to being developed for a specific task, these systems tend to fairly monolithic in that their components have fairly strict dependencies on each other, which make plug-and-play functionality difficult. NLP researchers and systems developers in the field realize that modularized approaches are beneficial for component reuse and more rapid development and advancement of NLP technology. In addition to the issue of modularity, the NLP systems development efforts are starting to take scalability into account. The Mayo Clinic's repository of clinical notes contains over 16 million documents growing at the rate of 50K documents per week. The time and space required for processing these large amounts of data impose constraints on the complexity of NLP systems.

Another engineering challenge is to make the NLP systems work in real time. This is particularly important in a clinical environment for patient recruitment or patient identification for clinical research use cases. In order to satisfy this requirement, a text processing system has to interface with the Electronic Health Record (EHR) system in real time and process documents immediately after they become available electronically. All of these are non-trivial issues and are currently being addressed in the community. In this poster we present the design and architecture of a large-scale, highly modularized, real-time enabled text analysis system as well as experimental validation results.

## 2 System Description

Mayo Clinic and IBM have collaborated on a Text Analytics project as part of a strategic Life Sciences and Computational Biology partnership. The goal of the Text Analytics collaboration was to provide a text analysis system that would index and retrieve clinical documents at the Mayo Clinic.

The Text Analytics architecture leveraged existing interface feeds for clinical documents by routing them to the warehouse. A work manager was written using messaging queues to distribute work for text analysis for real-time and bulk processing (see Figure 1). Additional text analysis engines can be configured and added with appropriate hardware to increase document throughput of the system.

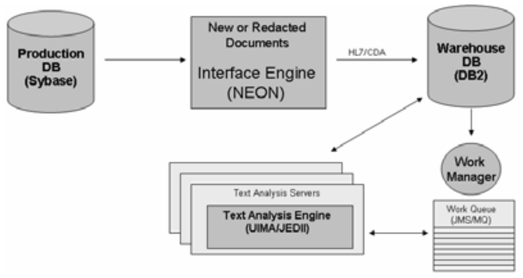


**Figure 1- Text Analysis Process Flow**

For deployment of text analysis engines we tested two configurations. During the development phase we used synchronous messaging using Apache Web Server with Tomcat/Axis. The Apache Web server provided a round robin mechanism to distributed SOAP requests for text analysis. This testing was deployed on a 20 CPU Beowulf cluster using AMD Athlon™ processors running Linux operating system. For production deployment we used Message Driven Beans (MDBs)using IBM Websphere Application Server™ (WAS) and IBM Websphere Message Queue™. The text engines were deployed on 2-CPU blade servers with 4Gb RAM. Each WAS instance had two MDBs with text analysis engines.

Work was distributed using message queues. Each text analysis engine was deployed to function independent of other engines. A total of 20 blade servers were configured for text processing. The average document throughput for each blade was 20 documents per minute.

The text analysis engine was designed by conceptually breaking up the task into granular functions that could be implemented as components to be assembled into a text processing system.

To implement the components we used an IBM AlphaWorks package called Unstructured Information Management Architecture (UIMA). UIMA is a software architecture that defines roles, interface, and communications of components for natural language processing. The four main UIMA services include: acquisition, unstructured information analysis, structured information access, and component discovery. For the Mayo project we used the first three services. The ability to customize annotator sequences was advantageous during the design process. Also, the ability to add annotators for specific dictionaries amounted only in minor work. Once annotators are written to conformance, UIMA provides pipeline development and permits the developer to quickly customize processing to a specific task. The final annotator layout is depicted in Figure 2.

The *context free tokenizer* is a finite state transducer that parses the document text into the smallest meaningful spans of text. A token is a set of characters that can be classified into one of these categories: word, punctuation, number, contraction, possessive, symbol without taking into account any additional context.

The *context sensitive spell corrector annotator* is used for automatic spell correction on word tokens. This annotator uses a combination of isolated-word and context-sensitive statistical approaches to rank the possible suggestions [5]. The suggestion with the highest ranking is stored as a feature of a token.

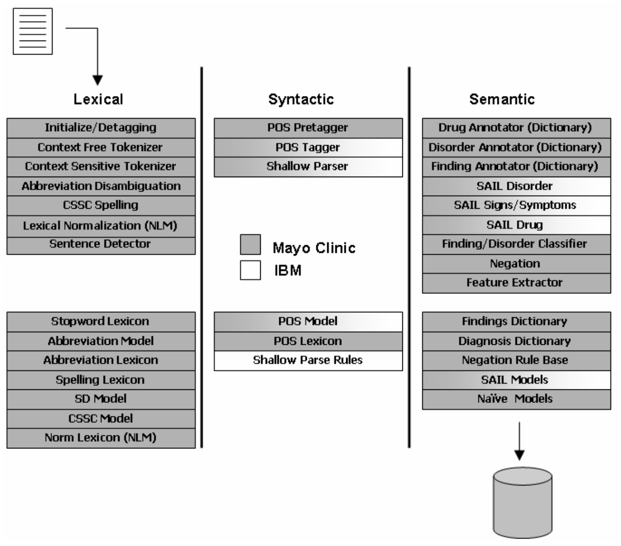


**Figure 2 – Text Analysis Pipeline**

The *lexical normalizer annotator* is applied only to words, possessives, and contractions. It generates a canonical form by using the National Library of Medicine UMLS Lexical Variant Generator (LVG) tool[1]. Apart from generating lexical variants and stemming optimized for the biomedical domain, it also generates a list of lemma entries with Penn Treebank tags as input for the POS tagger.

The *sentence detector annotator* parses the document text into sentences. The sentence detector is based on a Maximum Entropy classifier technology[2] and is trained to recognize sentence boundaries from hand annotated data.

[1] http://umlslex.nlm.nih.gov
[2] http://maxent.sourceforge.net/

The context dependent tokenizer uses context to detect complex tokens such as dates, times, and problem lists[3].

The *part of speech (POS) pre-tagger annotator* is intended to execute prior to the POS tagger annotator. The pre-tagger loads a list of words that are unambiguous with respect to POS and have predetermined Penn Treebank tags. Words in the document text are tagged with these predetermined tags. The POS tagger can ignore these words and focus on the remaining syntactically ambiguous words.

The *POS tagger annotator* attaches a part of speech tag to each token. The current version of the POS tagger is from IBM based on Hidden Markov models technology. This tagger has been trained on a combination of the Penn Treebank corpus of general English and a corpus of manually tagged clinical data developed at the Mayo Clinic [6], [7].

The *shallow parser annotator* makes higher level constructs at the phrase level. The Shallow Parser is from IBM. The shallow parser uses a set of rules operating on tokens and their part-of-speech category to identify linguistic phrases in the text such as noun phrases, verb phrases, and adjectival phrases.

The *dictionary named entity annotator* uses a set of enriched dictionaries (SNOMED-CT, MeSH, RxNorm and Mayo Synonym Clusters (MSC) to lookup named entities in the document text. These named entities include drugs, diagnoses, signs, and symptoms. The MSC database contains a set of clusters each consisting of diagnostic statements that are considered to be synonymous. Synonymy here is defined as two or more terms that have been manually classified to the same category in the Mayo Master Sheet repository, which contains over 20 million manually coded diagnostic statements. These diagnostic statements are used as entry terms for dictionary lookup. A set of Mayo compiled dictionaries are also used to detect abbreviations and hyphenated terms.

The *abbreviation disambiguation annotator* attempts to detect and expand abbreviations and acronyms based on Maximum Entropy classifiers trained on automatically generated data [8].

[3] Problem lists typically consist of numbered items in the Impression/Report/Plan section of the clinical notes

The *negation annotator* assigns a certainty attribute to each named entity with the exception of drugs. This annotator is based on a generalized version of Chapman's NegEx algorithm [9].

The *ML (Machine Learning) Named Entity annotator* is based on a Naïve Bayes classifier trained on a combination of the UMLS entry terms and the MCS where each diagnostic statement is represented as a bag-of-words and used as a training sample for generating a Naive Bayes classifier which assigns MCS id's to noun phrases identified in the text of clinical notes. The architecture of this component is given in Figure 3.

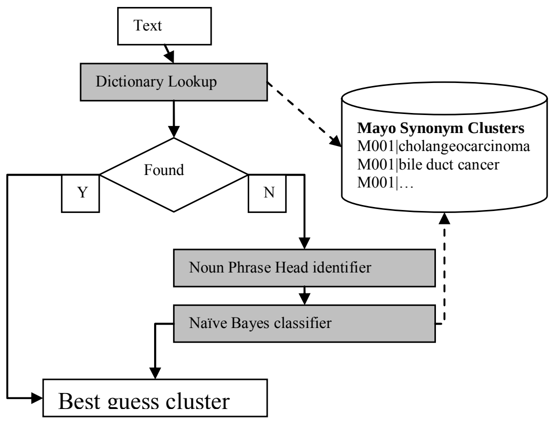


Figure 3. ML Named Entity Classifier

The text of a clinical note is first looked up in the MSC database using the *dictionary named entity annotator*. If a span of text matched something in the database, then the span is marked as a named entity annotation and the appropriate cluster ID is assigned to it. The portions of text where no match was found continue to be processed with a named entity identification algorithm that relies on the output of the *shallow parser annotator* to find noun phrases whose heads are on a list of nouns that exist in the MSC database as individual manually coded entries. For example, a noun phrase such as 'metastasized cholangiocarcinoma' will be identified as a named entity and subsequently automatically classified, but a noun phrase such as 'patient's father' will not.

## 3 Evaluation

The system performance was evaluated using a collection of 351 documents partitioned into 4 topics: pulmonary fibrosis, cholangiocarcinoma, diabetes mellitus and congestive heart failure. Each of the topics contained approximately 90 documents that were manually examined by three nurse abstractors and three physicians. Each note was marked as either relevant or not relevant to a given topic. In order to establish the reliability of this test corpus, we used a standard weighted Kappa statistic [10]. The overall Kappa for the four topics were 0.59 for pulmonary fibrosis, 0.79 for cholangiocarcinoma, 0.79 for diabetes mellitus and 0.59 for congestive heart failure. We ran a set of queries for each of the 4 topics on the partition generated for that topic. Each query used the primary term that represented the topic. For example, for pulmonary fibrosis, only the term 'pulmonary fibrosis' was used while other closely related terms such as 'interstitial pneumonitis' were excluded. The baseline query was executed using the term as a key phrase on the original text of the documents. The rest of the queries were executed using the concept id's automatically generated for each primary term. On the back end, the text of the clinical notes was annotated with the Metamap program [3] for the UMLS concepts and the *ML Named Entity annotator* for MSC cluster id's. On the front end, the UMLS concept id's were generated via the UMLS Knowledge Server online and the MSC id's were generated using a combination of the same Naïve Bayes classifier and the same dictionary lookup mechanism as were used to annotate the clinical notes. We also tested a query that combined Metamap and MSC annotations and query parameters. Recall, precision and f-score (α=0.5) were calculated for each query. The results are summarized in Table 1.

| | **Precision** | **Recall** | **F-score** |
|---|---|---|---|
| Key Phrase | 0.77 | 0.73 | 0.749467 |
| MSC cluster | 0.67 | 0.89 | 0.764487 |
| Metamap | 0.71 | 0.84 | 0.769548 |
| Metamap+MSC | 0.67 | **0.92** | 0.775346 |

Table 1. Performance of different annotation methods.

The f-score results are fairly close for all methods; however, the recall is highest for the method that combines Metamap and the MSC methodology. This is particularly important for using this system in recruiting patients for epidemiological research for disease incidence or disease prevalence studies and clinical trials where recall is valued more than precision. A combination of Metamap and MSC annotations and queries produced the highest recall which shows that these systems are complementary. The modular design of our system makes it easy to incorporate complementary annotation systems like Metamap into the annotation process.

**Acknowledgements**

The authors wish to thank the Mayo Clinic Emeritus Staff Physicians and Nurse Abstractors who served as experts for this study. The authors also wish to thank Patrick Duffy for programming support and David Hodge for statistical analysis and interpretation.

## References

1. Friedman, C., et al., *A general natural-language text processor for clinical radiology*. Journal of American Medical Informatics Association, 1994. **1**(2): p. 161-174.
2. Friedman, C. *Towards a Comprehensive Medical Language Processing System: Methods and Issues*. in *American Medical Informatics Association (AMIA)*. 1997.
3. Aronson, A. *Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program*. in *Proceedings of the 2001 AMIA Annual Symposium*. 2001. Washington, DC.
4. Mitchell, K. and R. Crowley. *GNegEx – Implementation and Evaluation of a Negation Tagger for the Shared Pathology Iinformatics Network*. in *Advancing Practice, Instruction and Innovation through Informatics (APIII)*. 2003.
5. Thompson-McInness, B., S. Pakhomov, and T. Pedersen. *Automating Spelling Correction Tools Using Bigram Statistics*. in *Medinfo Symposium*. 2004. San Francisco, CA, USA.
6. Coden, A., et al., *Domain-specific language models and lexicons for tagging*. In print in Journal of Biomedical Informatics, 2005.
7. Pakhomov, S., A. Coden, and C. Chute, *Developing a Corpus of Clinical Notes Manually Annotated for Part-of-Speech*. To appear in International Journal of Medical Informatics, 2005(Special Issue on Natural Language Processing in Biomedical Applications).
8. Pakhomov, S. *Semi-Supervised Maximum Entropy Based Approach to Acronym and Abbreviation Normalization in Medical Texts*. in *40th Meeting of the Association for Computational Linguistics (ACL 2002)*. 2002. Philadelohia, PA.
9. Chapman, W.W., et al. *Evaluation of Negation Phrases in Narrative Clinical Reports*. in *American Medical Informatics Association*. 2001. Washington, DC, USA.
10. Landis, J.R. and G.G. Koch, *The Measurement of Observer Agreement for Categorical Data*. Biometrics, 1977. **33**: p. 159-174.

# A Voice Enabled Procedure Browser for the International Space Station

**Manny Rayner, Beth Ann Hockey, Nikos Chatzichrisafis, Kim Farrell**
ICSI/UCSC/RIACS/NASA Ames Research Center
Moffett Field, CA 94035–1000
mrayner@riacs.edu, bahockey@email.arc.nasa.gov
Nikos.Chatzichrisafis@web.de, kfarrell@email.arc.nasa.gov

**Jean-Michel Renders**
Xerox Research Center Europe
6 chemin de Maupertuis, Meylan, 38240, France
Jean-Michel.Renders@xrce.xerox.com

## Abstract

Clarissa, an experimental voice enabled procedure browser that has recently been deployed on the International Space Station (ISS), is to the best of our knowledge the first spoken dialog system in space. This paper gives background on the system and the ISS procedures, then discusses the research developed to address three key problems: grammar-based speech recognition using the Regulus toolkit; SVM based methods for open microphone speech recognition; and robust side-effect free dialogue management for handling undos, corrections and confirmations.

## 1 Overview

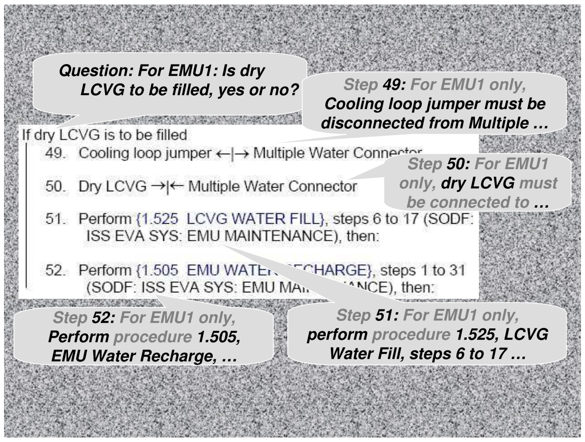
Astronauts on the International Space Station (ISS) spend a great deal of their time performing complex procedures. Crew members usually have to divide their attention between the task and a paper or PDF display of the procedure. In addition, since objects float away in microgravity if not fastened down, it would be an advantage to be able to keep both eyes and hands on the task. Clarissa, an experimental speech enabled procedure navigator (Clarissa, 2005), is designed to address these problems. The system was deployed on the ISS on January 14, 2005 and is scheduled for testing later this year; the initial version is equipped with five XML-encoded procedures, three for testing water quality and two for space suit maintenance. To the best of our knowledge, Clarissa is the first spoken dialogue application in space.

The system includes commands for navigation: forward, back, and to arbitrary steps. Other commands include setting alarms and timers, recording, playing and deleting voice notes, opening and closing procedures, querying system status, and inputting numerical values. There is an optional mode that aggressively requests confirmation on completion of each step. Open microphone speech recognition is crucial for providing hands free use. To support this, the system has to discriminate between speech that is directed to it and speech that is not. Since speech recognition is not perfect, and additional potential for error is added by the open microphone task, it is also important to support commands for undoing or correcting bad system responses.

The main components of the Clarissa system are a speech recognition module, a classifier for executing the open microphone accept/reject decision, a semantic analyser, and a dialogue manager. The rest of this paper will briefly give background on the structure of the procedures and the XML representation, then describe the main research content of the system.

## 2 Voice-navigable procedures

ISS procedures are formal documents that typically represent many hundreds of person hours of preparation, and undergo a strict approval process. One requirement in the Clarissa project was that the procedures should be displayed visually exactly as they


Question: For EMU1: Is dry LCVG to be filled, yes or no?
Step 49: For EMU1 only, Cooling loop jumper must be disconnected from Multiple …
Step 50: For EMU1 only, dry LCVG must be connected to …
Step 52: For EMU1 only, Perform procedure 1.505, EMU Water Recharge, …
Step 51: For EMU1 only, perform procedure 1.525, LCVG Water Fill, steps 6 to 17 …

If dry LCVG is to be filled
49. Cooling loop jumper ←|→ Multiple Water Connector
50. Dry LCVG →|← Multiple Water Connector
51. Perform {1.525 LCVG WATER FILL}, steps 6 to 17 (SODF: ISS EVA SYS: EMU MAINTENANCE), then:
52. Perform {1.505 EMU WATER RECHARGE}, steps 1 to 31 (SODF: ISS EVA SYS: EMU MAINTENANCE), then:

Figure 1: Adding voice annotations to a group of steps

appear in the original PDF form. However, reading these procedures verbatim would not be very useful. The challenge is thus to let the spoken version diverge significantly from the written one, yet still be similar enough in meaning that the people who control the procedures can be convinced that the two versions are in practice equivalent.

Figure 1 illustrates several types of divergences between the written and spoken versions, with "speech bubbles" showing how procedure text is actually read out. In this procedure for space suit maintenance, one to three suits can be processed. The group of steps shown cover filling of a "dry LCVG". The system first inserts a question to ask which suits require this operation, and then reads the passage once for each suit, specifying each time which suit is being referred to; if no suits need to be processed, it jumps directly to the next section. Step 51 points the user to a subprocedure. The spoken version asks if the user wants to execute the steps of the subprocedure; if so, it opens the LCVG Water Fill procedure and goes directly to step 6. If the user subsequently goes past step 17 of the subprocedure, the system warns that the user has gone past the required steps, and suggests that they close the procedure.

Other important types of divergences concern entry of data in tables, where the system reads out an appropriate question for each table cell, confirms the value supplied by the user, and if necessary warns about out-of-range values.

| Rec | Patterns | Errors | | |
|---|---|---|---|---|
| | | Reject | Bad | Total |
| Text | LF | 3.1% | 0.5% | 3.6% |
| Text | Surface | 2.2% | 0.8% | 3.0% |
| Text | Surface+LF | 0.8% | 0.8% | 1.6% |
| SLM | Surface | 2.8% | 7.4% | 10.2% |
| GLM | LF | 1.4% | 4.9% | 6.3% |
| GLM | Surface | 2.9% | 4.8% | 7.7% |
| **GLM** | **Surface+LF** | **1.0%** | **5.0%** | **6.0%** |

Table 1: Speech understanding performance on six different configurations of the system.

## 3 Grammar-based speech understanding

Clarissa uses a grammar-based recognition architecture. At the start of the project, we had two main reasons for choosing this approach over the more popular statistical one. First, we had no available training data. Second, the system was to be designed for experts who would have time to learn its coverage, and who moreover, as former military pilots, were comfortable with the idea of using controlled language. Although there is not much to be found in the literature, an earlier study in which we had been involved (Knight et al., 2001) suggested that grammar-based systems outperformed statistical ones for this kind of user. Given that neither of the above arguments is very strong, we wanted to implement a framework which would allow us to compare grammar-based methods with statistical ones, and retain the option of switching from a grammar-based framework to a statistical one if that later appeared justified. The Regulus and Alterf platforms, which we have developed under Clarissa and other earlier projects, are designed to meet these requirements.

The basic idea behind Regulus (Regulus, 2005; Rayner et al., 2003) is to extract grammar-based language models from a single large unification grammar, using example-based methods driven by small corpora. Since grammar construction is now a corpus-driven process, the same corpora can be used to build statistical language models, facilitating a direct comparison between the two methodologies.

On its own, however, Regulus only permits comparison at the level of recognition strings. Alterf (Rayner and Hockey, 2003) extends the paradigm to

| ID | Rec | Features | Classifier | Error rates | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Classification | | | | Task |
| | | | | In domain | | Out | Av | |
| | | | | Good | Bad | | | |
| 1 | SLM | Confidence | Threshold | 5.5% | 59.1% | 16.5% | 11.8% | 10.1% |
| 2 | GLM | Confidence | Threshold | 7.1% | 48.7% | 8.9% | 9.4% | 7.0% |
| 3 | SLM | Confidence + Lexical | Linear SVM | 2.8% | 37.1% | 9.0% | 6.6% | 7.4% |
| 4 | GLM | Confidence + Lexical | Linear SVM | 2.8% | 48.5% | 8.7% | 6.3% | 6.2% |
| 5 | SLM | Confidence + Lexical | Quadratic SVM | 2.6% | 23.6% | 8.5% | 5.5% | 6.9% |
| 6 | **GLM** | **Confidence + Lexical** | **Quadratic SVM** | **4.3%** | **28.1%** | **4.7%** | **5.5%** | **5.4%** |

Table 2: Performance on accept/reject classification and the top-level task, on six different configurations.

the semantic level, by providing a trainable semantic interpretation framework. Interpretation uses a set of user-specified patterns, which can match either the surface strings produced by both the statistical and grammar-based architectures, or the logical forms produced by the grammar-based architecture.

Table 1 presents the result of an evaluation, carried out on a set of 8158 recorded speech utterances, where we compared the performance of a statistical/robust architecture (SLM) and a grammar-based architecture (GLM). Both versions were trained off the same corpus of 3297 utterances. We also show results for text input simulating perfect recognition. For the SLM version, semantic representations are constructed using only surface Alterf patterns; for the GLM and text versions, we can use either surface patterns, logical form (LF) patterns, or both. The "Error" columns show the proportion of utterances which produce no semantic interpretation ("Reject"), the proportion with an incorrect semantic interpretation ("Bad"), and the total.

Although the WER for the GLM recogniser is only slightly better than that for the SLM recogniser (6.27% versus 7.42%, 15% relative), the difference at the level of semantic interpretation is considerable (6.3% versus 10.2%, 39% relative). This is most likely accounted for by the fact that the GLM version is able to use logical-form based patterns, which are not accessible to the SLM version. Logical-form based patterns do not appear to be intrinsically more accurate than surface (contrast the first two "Text" rows), but the fact that they allow tighter integration between semantic understanding and language modelling is intuitively advantageous.

## 4 Open microphone speech processing

The previous section described speech understanding performance in terms of correct semantic interpretation of in-domain input. However, open microphone speech processing implies that some of the input will not be in-domain. The intended behaviour for the system is to reject this input. We would also like it, when possible, to reject in-domain input which has not been correctly recognised.

Surface output from the Nuance speech recogniser is a list of words, each tagged with a confidence score; the usual way to make the accept/reject decision is by using a simple threshold on the average confidence score. Intuitively, however, we should be able to improve the decision quality by also taking account of the information in the recognised words.

By thinking of the confidence scores as weights, we can model the problem as one of classifying documents using a weighted bag of words model. It is well known (Joachims, 1998) that Support Vector Machine methods are very suitable for this task. We have implemented a version of the method described by Joachims, which significantly improves on the naive confidence score threshold method.

Performance on the accept/reject task can be evaluated directly in terms of the classification error. We can also define a metric for the overall speech understanding task which includes the accept/reject decision, as a weighted loss function over the different types of error. We assign weights of 1 to a false reject of a correct interpretation, 2 to a false accept of an incorrectly interpreted in-domain utterance, and 3 to a false accept of an out-of-domain utterance. This captures the intuition that correcting false accepts is considerably harder than correcting false rejects, and that false accepts of utterances not directed at the system are worse than false accepts of incorrectly interpreted utterances.

Table 2 summarises the results of experiments comparing performance of different recognisers and accept/reject classifiers on a set of 10409 recorded utterances. "GLM" and "SLM" refer respectively to the best GLM and SLM recogniser configurations from Table 1. "Av" refers to the average classifier error, and "Task" to a normalised version of the weighted task metric. The best SVM-based method (line 6) outperforms the best naive threshold method (line 2) by 5.4% to 7.0% on the task metric, a relative improvement of 23%. The best GLM-based method (line 6) and the best SLM-based method (line 5) are equally good in terms of accept/reject classification accuracy, but the GLM's better speech understanding performance means that it scores 22% better on the task metric. The best quadratic kernel (line 6) outscores the best linear kernel (line 4) by 13%. All these differences are significant at the 5% level according to the Wilcoxon matched-pairs test.

## 5 Side-effect free dialogue management

In an open microphone spoken dialogue application like Clarissa, it is particularly important to be able to undo or correct a bad system response. This suggests the idea of representing discourse states as objects: if the complete dialogue state is an object, a move can be undone straightforwardly by restoring the old object. We have realised this idea within a version of the standard "update semantics" approach to dialogue management (Larsson and Traum, 2000); the whole dialogue management functionality is represented as a declarative "update function" relating the old dialogue state, the input dialogue move, the new dialogue state and the output dialogue actions.

In contrast to earlier work, however, we include task information as well as discourse information in the dialogue state. Each state also contains a backpointer to the previous state. As explained in detail in (Rayner and Hockey, 2004), our approach permits a very clean and robust treatment of undos, corrections and confirmations, and also makes it much simpler to carry out systematic regression testing of the dialogue manager component.

## Acknowledgements

Work at ICSI, UCSC and RIACS was supported by NASA Ames Research Center internal funding. Work at XRCE was partly supported by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Several people not credited here as coauthors also contributed to the implementation of the Clarissa system: among these, we would particularly like to mention John Dowding, Susana Early, Claire Castillo, Amy Fischer and Vladimir Tkachenko. This publication only reflects the authors' views.

## References

Clarissa, 2005. http://www.ic.arc.nasa.gov/projects/clarissa/. As of 26 April 2005.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, Chemnitz, Germany.

S. Knight, G. Gorrell, M. Rayner, D. Milward, R. Koeling, and I. Lewin. 2001. Comparing grammar-based and robust approaches to speech understanding: a case study. In *Proceedings of Eurospeech 2001*, pages 1779–1782, Aalborg, Denmark.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering, Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering*, pages 323–340.

M. Rayner and B.A. Hockey. 2003. Transparent combination of rule-based and data-driven approaches in a speech understanding architecture. In *Proceedings of the 10th EACL (demo track)*, Budapest, Hungary.

M. Rayner and B.A. Hockey. 2004. Side effect free dialogue management in a voice enabled procedure browser. In *Proceedings of INTERSPEECH 2004*, Jeju Island, Korea.

M. Rayner, B.A. Hockey, and J. Dowding. 2003. An open source environment for compiling typed unifi cation grammars into speech recognisers. In *Proceedings of the 10th EACL*, Budapest, Hungary.

Regulus, 2005. http://sourceforge.net/projects/regulus/. As of 26 April 2005.

# The Linguist's Search Engine: An Overview

**Philip Resnik**
Department of Linguistics and UMIACS
University of Maryland
College Park, MD 20742
`resnik@umd.edu`

**Aaron Elkiss**
UMIACS
University of Maryland
College Park, MD 20742
`aelkiss@umiacs.umd.edu`

## Abstract

The Linguist's Search Engine (LSE) was designed to provide an intuitive, easy-to-use interface that enables language researchers to seek linguistically interesting examples on the Web, based on syntactic and lexical criteria. We briefly describe its user interface and architecture, as well as recent developments that include LSE search capabilities for Chinese.

## 1 Introduction

The idea for the Linguist's Search Engine originated in a simple frustration shared by many people who study language: the fact that so much of the argumentation in linguistic theory is based on subjective judgments. Who among us has not, in some talk or class, heard an argument based on a "starred" (deemed-ungrammatical) example, and whispered to someone nearby, *Did that sound ok to you?* because we thought it sounded fine? As Bard et al. (1996) put it, each linguistic judgment is a "small and imperfect experiment"'. Schütze (1996) and Cowart (1997) provide detailed discussion of instability and unreliability in such informal methods, which can lead to biased or even misleading results.

Recent work on linguistics methodology draws on the perception literature in psychology to provide principled methods for eliciting gradient, rather than discrete, linguistic judgments (Sorace and Keller, 2005). In addition, at least as far back as Rich Pito's 1992 *tgrep*, distributed with the Penn Treebank, computationally sophisticated linguists have had the option of looking at naturally occurring data rather than relying on constructed sentences and introspective judgments (e.g., Christ, 1994; Corley et al., 2001; Blaheta, 2002; Kehoe and Renouf 2002; König and Lezius, 2002; Fletcher 2002; Kilgarriff 2003). Unfortunately, many linguists are unwilling to invest in psycholinguistic methods, or in the computational skills necessary for working with corpus search tools. A variety of people interested in language have moved in the direction of using Web search engines such as Google as a source of naturally occurring data, but conventional search engines do not provide the mechanisms needed to perform many of the simplest linguistically informed searches – e.g., seeking instances of a particular verb used only intransitively.

The Linguist's Search Engine (LSE) was designed to provide the broadest possible range of users with an intuitive, linguistically sophisticated but user-friendly way to search the Web for naturally occurring data. Section 2 lays out the LSE's basic interface concepts via several illustrative examples. Section 3 discusses its architecture and implementation. Section 4 discusses the current status of the LSE and recent developments.

## 2 LSE Interface Concepts

The design of the LSE was guided by a simple basic premise: a tool can't be a success unless people use it. This led to the following principles in its design:

- Minimize learning/ramp-up time.
- Have a linguist-friendly look and feel.
- Permit rapid interaction.
- Permit large-scale searches.
- Allow searches using linguistic criteria.

Some of these principles conflict with each other. For example, sophisticated searches are difficult to specify in a linguist-friendly way and without requiring some learning by the user, and rapid interaction is difficult to accomplish for Web-sized searches.

## 2.1 Query By Example

The LSE adopts a strategy one can call "query by example," in order to provide sophisticated search functionality without requiring the user to learn a complex query language. For example, consider the so-called "comparative correlative" construction (Culicover and Jackendoff, 1999). Typing *the bigger the house the richer the buyer* automatically produces the analysis in Figure 1, which can be edited with a few mouse clicks to get the generalized structure in Figure 2, converted with one button push into the LSE's query language, and then submitted in order to find other examples of this construction, such as *The higher the rating, the lower the interest rate that must be paid to investors; The more you bingo, the more chances you have in the drawing; The more we plan and prepare, the easier the transition.*

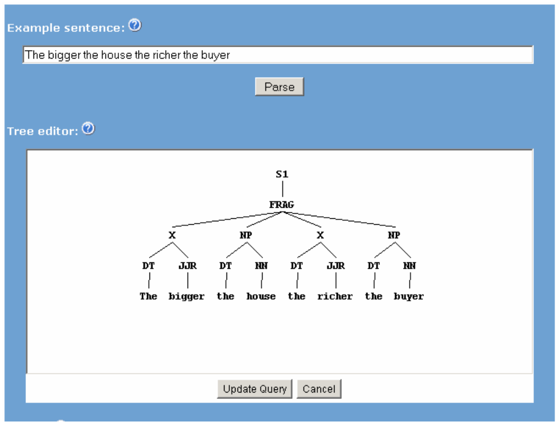


***Figure 1. Querying by example***

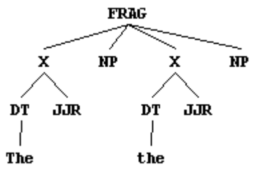


***Figure 2. Generalized query***

Crucially, users need not learn a query language, although advanced users can edit or create queries directly if so desired. Nor do users need to agree with (or even understand) the LSE's automatic parse, in order to find sentences with parses similar to the exemplar. Indeed, as is the case in Figure 1, the parse need not even be entirely reasonable; what is important is that the structure produced when analyzing the query will be the *same* structure produced via analysis of the corresponding sentences in the corpus.

Other search features include the ability to specify immediate versus non-immediate dominance; the ability to negate relationships (e.g. a VP that does *not* immediately dominate an NP); the ability to specify that words should match on all morphological forms; the ability to match nodes based on WordNet relationships (e.g. all descendants of a particular word sense); the ability to save and reload queries; the ability to download results in keyword-in-context (KWIC) format; and the ability to apply a simple keyword-based filter to avoid offensive results during live demonstrations.

Results are typically returned by the LSE within a few seconds, in a simple search-engine style format. In addition, however, the user has rapid access to the immediate preceding and following contexts of returned sentences, their annotations, and the Web page where the example occurred.

## 2.2 Built-In and Custom Collections

Linguistically annotating and indexing the entire Web is beyond impractical, and therefore there is a clear tradeoff between rapid response time and the ability to search the Web as a whole. In order to manage this tradeoff, the LSE provides, by default, a built-in collection of English sentences taken randomly from a Web-scale crawl at the Internet

Archive.[1] This static collection is often useful by itself.

In order to truly search the entire Web, the LSE permits users to define their own custom collections, piggybacking on commercial Web search engines. Consider, as an example, a search involving the verb *titrate*, which is rare enough that it occurs only twice in a collection of millions of sentences. Using the LSE's "Build Custom Collection" functionality, the user can specify that the LSE should:

- Query Altavista to find pages containing any morphological form of *titrate*
- Extract only sentences containing that verb
- Annotate and index those sentences
- Augment the collection by iterating this process with different specifications

Doing the Altavista query and extracting, parsing, and indexing the sentences can take some time, but the LSE permits the user to begin searching his or her custom collection as soon as *any* sentences have been added into it. Typically dozens to hundreds of sentences are available within a few minutes, and a typical custom collection, containing thousands or tens of thousands of sentences, is completed within a few hours. Collections can be named, saved, augmented, and deleted.

Currently the LSE supports custom collections built using searches on Altavista and Microsoft's MSN Search. It is interesting to note that the search engines' capabilities can be used to create custom collections based on extralinguistic criteria; for example, specifying pages originating only in the *.uk* domain in order to increase the likelihood of finding British usages, or specifying additional query terms in order to bias the collection toward particular topics or domains.

## 3 Architecture and Implementation

The LSE's design can be broken into the following high level components:

- User interface
- Search engine interface
- NLP annotation
- Indexing
- Search

The design is centered on a relational database that maintains information about users, collections, documents, and sentences, and the implementation combines custom-written code with significant use of off-the-shelf packages. The interface with commercial search engines is accomplished straightforwardly by use of the WWW::Search perl module (currently using a custom-written variant for MSN Search).

Natural language annotation is accomplished via a parallel, database-centric annotation architecture (Elkiss, 2003). A configuration specification identifies dependencies between annotation tasks (e.g. tokenization as a prerequisite to part-of-speech tagging). After documents are processed to handle markup and identify sentence boundaries, individual sentences are loaded into a central database that holds annotations, as well as information about which sentences remain to be annotated. Crucially, sentences can be annotated in parallel by task processes residing on distributed nodes.

Indexing and search of annotations is informed by the recent literature on semistructured data. However, linguistic databases are unlike most typical semistructured data sets (e.g., sets of XML documents) in a number of respects – these include the fact that the dataset has a very large schema (tens of millions of distinct paths from root node to terminal symbols), long path lengths, a need for efficient handling of queries containing wildcards, and a requirement that all valid results be retrieved. On the other hand, in this application incremental updating is not a requirement, and neither is 100% precision: results can be overgenerated and then filtered using a less efficient comparison tools such as *tgrep2*. Currently the indexing scheme follows ViST (Wang et al., 2003), an approach based on suffix trees that indexes structure and content together. The variant implemented in the LSE ignores insufficiently selective query branches, and achieves more efficient search by modifying the ordering within the structural index, creating an in-memory tree for the query, ordering processing of

[1] The built-in LSE Web collection contains 3 million sentences at the time of this writing. We estimate that it can be increased by an order of magnitude without seriously degrading response time, and we expect to do so by the time of the demonstration.

query branches from most to least selective, and memoizing query subtree matches.

## 4 Status and Recent Developments

The LSE "went live" on January 20, 2004 and approximately 1000 people have registered and tried at least one query. In response to a recent survey, several dozen LSE users reported having tried it more than casually, and there are a dozen or so reports of the LSE having proven useful in real work, either for research or as a tool that was useful in teaching. Resnik et al. (2005) describe two pieces of mainstream linguistics research – one in psycholinguistics and one in theoretical syntax – in which the LSE played a pivotal role.

The LSE software is currently being documented and packaged up, for an intended open-source release.[2] In addition to continuing linguistic research with the LSE, we are also experimenting with alternative indexing/search schemes. Finally, we are engaged in a project adapting the LSE for use in language pedagogy – specifically, as a tool assisting language teaching specialists in creating training and testing materials for learners of Chinese. For that purpose, we are experimenting with a built-in collection of Chinese Web documents that includes links to their English translations (Resnik and Smith, 2003).

## Acknowledgments

This work has received support from the National Science Foundation under ITR grant IIS01130641, and from the Center for the Advanced Study of Language under TTO32. The authors are grateful to Christiane Fellbaum and Mari Broman Olsen for collaboration and discussions; to Rafi Khan, Saurabh Khandelwal, Jesse Metcalf-Burton, G. Craig Murray, Usama Soltan, and James Wren for their contributions to LSE development; and to Doug Rohde, Eugene Charniak, Adwait Ratnaparkhi, Dekang Lin, UPenn's XTAG group, Princeton's WordNet project, and untold others for software components used in this work.

## References

Bard, E.G., Robertson, D. and A. Sorace. Magnitude estimation of linguistic acceptability. *Language* 72.1: 32-68, 1996.

Christ, Oli. A modular and flexible architecture for an integrated corpus query system, COMPLEX'94, Budapest, 1994.

Corley, Steffan, Martin Corley, Frank Keller, Matthew W. Crocker, and Shari Trewin. Finding Syntactic Structure in Unparsed Corpora: The Gsearch Corpus Query System, *Computers and the Humanities*, 35:2, 81-94, 2001.

Cowart, Wayne. *Experimental Syntax: Applying Objective Methods to Sentence Judgments*, Sage Publications, Thousand Oaks, CA, 1997.

Culicover, Peter and Ray Jackendoff. The view from the periphery: the English comparative correlative. *Linguistic Inquiry* 30:543-71, 1999.

Elkiss, Aaron. A Scalable Architecture for Linguistic Annotation. Computer Science Undergraduate Honors Thesis. University of Maryland. May 2003.

Fletcher, William. Making the Web More Useful as a Source for Linguistic Corpora, North American Symposium on Corpus Linguistics, 2002.

Kehoe, Andrew and Antoinette Renouf, WebCorp: Applying the Web to linguistics and linguistics to the Web, in *Proceedings of WWW2002*, Honolulu, Hawaii, 7-11 May 2002.

Adam Kilgarriff, Roger Evans, Rob Koeling, David Tugwell. WASPBENCH: a lexicographer's workbench incorporating state-of-the-art word sense disambiguation. *Proceedings of EACL 2003*, 211-214, 2003.

Koenig, Esther and Lezius, Wolfgang, A description language for syntactically annotated corpora. In: *Proceedings of the COLING Conference*, pp. 1056-1060, Saarbruecken, Germany, 2002.

Schuetze, Carson. *The Empirical Base of Linguistics*, University of Chicago Press, 1996.

Sorace, Antonella and Frank Keller. Gradience in Linguistic Data. To appear in *Lingua*, 2005.

Philip Resnik and Noah A. Smith, The Web as a Parallel Corpus, *Computational Linguistics* 29(3), pp. 349-380, September 2003.

Philip Resnik, Aaron Elkiss, Ellen Lau, and Heather Taylor. The Web in Theoretical Linguistics Research: Two Case Studies Using the Linguist's Search Engine. 31st Meeting of the Berkeley Linguistics Society, February 2005.

H Wang, S Park, W Fan, and P Yu. ViST: a dynamic index method for querying XML data by tree structures. ACM SIGMOD 2003. pp. 110-121.

[2] Documentation maintained at http://lse.umiacs.umd.edu/.

# Learning Source-Target Surface Patterns for Web-based Terminology Translation

**Jian-Cheng Wu**
Department of Computer Science
National Tsing Hua University
101, Kuangfu Road,
Hsinchu, 300, Taiwan
`D928322@oz.nthu.edu.tw`

**Tracy Lin**
Dep. of Communication Eng.
National Chiao Tung University
1001, Ta Hsueh Road,
Hsinchu, 300, Taiwan
`tracylin@cm.nctu.edu.tw`

**Jason S. Chang**
Department of Computer Science
National Tsing Hua University
101, Kuangfu Road,
Hsinchu, 300, Taiwan
`jschang@cs.nthu.edu.tw`

## Abstract

This paper introduces a method for learning to find translation of a given source term on the Web. In the approach, the source term is used as a query and part of patterns to retrieve and extract translations in Web pages. The method involves using a bilingual term list to learn source-target surface patterns. At runtime, the given term is submitted to a search engine then the candidate translations are extracted from the returned summaries and subsequently ranked based on the surface patterns, occurrence counts, and transliteration knowledge. We present a prototype called *TermMine* that applies the method to translate terms. Evaluation on a set of encyclopedia terms shows that the method significantly outperforms the state-of-the-art online machine translation systems.

## 1 Introduction

Translation of terms has long been recognized as the bottleneck of translation by translators. By reusing prior translations a significant time spent in translating terms can be saved. For many years now, Computer-Aided Translation (CAT) tools have been touted as very useful for productivity and quality gains for translators. CAT tools such as Trados typically require up-front investment to populate multilingual terminology and translation memory. However, such investment has proven prohibitive for many in-house translation departments and freelancer translators and the actual productivity gains realized have been insignificant except for a few, very repetitive types of content. Much more productivity gain could be achieved by providing translation service of terminology.

Consider the job of translating a textbook such as "Artificial Intelligence – A Modern Approach." The best practice is probably to start by translating the indexes (Figure 1). It is not uncommon for these repetitive terms to be translated once and applied consistently throughout the book. For example, A good translation *F* = "聲學模型" for the given term *E* = "acoustic model," might be available on the Web due to the common practice of including the source terms (often in brackets, see Figure 2) when using a translated term (e.g. "**…**訓練出語音聲學模型（Acoustic Model） 及語言模型 **…**")**.** The surface patterns of co-occurring source and target terms (e.g., "*F*（*E*") can be learned by using the Web as corpus. Intuitively, we can submit *E* and *F* to a search engine

Figure 1. Some index entries in "Artificial intelligence – A Modern Approach" page 1045.

academy award, 458
accessible, 41
accusative case, 806
Acero, A., 580, 1010
Acharya, A., 131, 994
achieves, 389
Ackley, D. H., 133, 987
acoustic model, 568

Figure 2. Examples of web page summaries with relevant translations returned by Google for some source terms in Figure 1.

**1.** **...** 奧斯卡獎 Academy Awards. 柏林影展 Berlin International Film Festival. **...**
2. **...** 有兩個「固有格位」(inherent Case)，比如一個賓格 (accusative Case)、一個與 **...**
**3.** **...** 有一天，當艾克禮牧師(Alfred H. Ackley) 領完佈道會之後，有一猶太青年來問艾牧師說**..**
4. **..**語音辨識首先 先藉由大量的語料，求取其特徵參數，訓練出語音聲學模型（Acoustic Model）及語言模型**...**

and then extract the strings beginning with *F* and ending with *E* (or vice versa) to obtain recurring source-target patterns. At runtime, we can submit *E* as query, request specifically for target-language web-pages. With these surface patterns, we can then extract translation candidates *F*s from the summaries returned by the search engine. Additional information of occurrence counts and transliteration patterns can be taken into consideration to rank *F*s.

Table 1. Translations by the machine translation system *Google Translate* and *TermMine*.

| **Terms** | ***Google Translate*** | ***TermMine*** |
|---|---|---|
| academy award | *學院褒獎 | 奧斯卡獎 |
| accusative case | *對格案件 | 賓格 |
| Ackley | - | 艾克禮 |
| acoustic model | *音響模型 | 聲學模型 |

For instance, among many candidate translations, we will pick the translations "聲學模型" for "acoustic model" and "艾克禮" for "Ackley, " because they fit certain surface-target surface patterns and appears most often in the relevant webpage summaries. Furthermore, the first morpheme "艾" in "艾克禮" is consistent with prior transliterations of "A-" in "Ackley" (See Table 1).

We present a prototype system called *TermMine*, that automatically extracts translation on the Web (Section 3.3) based on surface patterns of target translation and source term in Web pages automatically learned on bilingual terms (Section 3.1). Furthermore, we also draw on our previous work on machine transliteration (Section 3.2) to provide additional evidence. We evaluate *TermMine* on a set of encyclopedia terms and compare the quality of translation of *TermMine* (Section 4) with a online translation system. The results seem to indicate the method produce significantly better results than previous work.

## 2 Related Work

There is a resurgent of interested in data-intensive approach to machine translation, a research area started from 1950s. Most work in the large body of research on machine translation (Hutchins and Somers, 1992), involves production of sentence-by-sentence translation for a given source text. In our work, we consider a more restricted case where the given text is a short phrase of terminology or proper names (e.g., “acoustic model” or “George Bush”).

A number of systems aim to translate words and phrases out of the sentence context. For example, Knight and Graehl (1998) describe and evaluate a multi-stage method for performing backwards transliteration of Japanese names and technical terms into English by the machine using a generative model. In addition, Koehn and Knight (2003) show that it is reasonable to define noun phrase translation without context as an independent MT subtask and build a noun phrase translation subsystem that improves statistical machine translation methods.

Nagata, Saito, and Suzuki (2001) present a system for finding English translations for a given Japanese technical term by searching for mixed Japanese-English texts on the Web. The method involves locating English phrases near the given Japanese term and scoring them based on occurrence counts and geometric probabilistic function of byte distance between the source and target terms. Kwok also implemented a term translation system for CLIR along the same line.

Cao and Li (2002) propose a new method to translate base noun phrases. The method involves first using Web-based method by Nagata et al., and if no translations are found on the Web, backing off to a hybrid method based on dictionary and Web-based statistics on words and context vectors. They experimented with noun-noun NP report that 910 out of 1,000 NPs can be translated with an average precision rate of 63%.

In contrast to the previous research, we present a system that automatically learns surface patterns for finding translations of a given term on the Web without using a dictionary. We exploit the convention of including the source term with the translation in the form of recurring patterns to extract translations. Additional evident of data redundancy and transliteration patterns is utilized to validate translations found on the Web.

## 3 The *TermMine* System

In this section we describe a strategy for searching the Web pages containing translations of a given term (e.g., “Bill Clinton” or “aircraft carrier”) and extracting translations therein. The proposed method involves learning the surface pattern knowledge (Section 3.1) necessary for locating translations. A transliteration model automatically trained on a list of proper name and transliterations (Section 3.2) is also utilized to evaluate and select transliterations for proper-name terms. These knowledge sources are used in concert to search, rank, and extract translations (Section 3.3).

## 3.1 Source and Target Surface patterns

With a set of terms and translations, we can learn the co-occurring patterns of a source term *E* and its translation *F* following the procedure below:

(1) Submit a conjunctive query (i.e. *E* AND *F*) for each pair (*E*, *F*) in a bilingual term list to a search engine.

(2) Tokenize the retrieved summaries into three types of tokens: I. A punctuation II. A source word, designated with the letter "w" III. A maximal block of target words (or characters in the case of language without word delimiters such as Mandarin or Japanese).

(3) Replace the tokens for *E*'s instances with the symbol "*E*" and the type-III token containing the translation *F* with the symbol "*F*". Note the token denoted as "*F*" is a maximal string covering the given translation but containing no punctuations or words in the source language.

(4) Calculate the distance between *E* and *F* by counting the number of tokens in between.

(5) Extract the strings of tokens from *E* to *F* (or the other way around) within a maximum distance of *d* (*d* is set to 3) to produce ranked surface patterns *P*.

For instance, with the source-target pair ("California," "加州") and a retrieved summary of "...亞州簡介. 北加州 Northern California. ...," the surface pattern "FwE" of distance 1 will be derived.

## 3.2 Transliteration Model

TermMine also relies on a machine transliteration model (Lin, Wu and Chang 2004) to confirm the transliteration of proper names. We use a list of names and transliterations to estimate the transliteration probability function P($\tau$| $\omega$), for any given transliteration unit (TU) $\omega$ and transliteration character (TC) $\tau$. Based on the Expectation Maximization (EM) algorithm. A TU for an English name can be a syllable or consonants which corresponds to a character in the target transliteration. Table 2 shows some examples of sub-lexical alignment between proper names and transliterations.

Table 2. Examples of aligned transliteration units.

| *Name* | *transliteration* | Viterbi alignment |
|---|---|---|
| `Spagna` | 斯帕尼亞 | `s-斯 pag-帕 n-尼 a-亞` |
| `Kohn` | 孔恩 | `Koh-孔 n-恩` |
| `Nayyar` | 納雅 | `Nay-納 yar-雅` |
| `Rivard` | 里瓦德 | `ri-里 var-瓦 d-德` |
| `Hall` | 霍爾 | `ha-霍 ll-爾` |
| `Kalam` | 卡藍 | `ka-卡 lam-藍` |

Figure 3. Transliteration probability trained on 1,800 bilingual names (λ denotes an empty string).

| τ | $\omega$ | P(τ\|$\omega$) | τ | $\omega$ | P(τ\|$\omega$) | τ | $\omega$ | P(τ\|$\omega$) |
|---|---|---|---|---|---|---|---|---|
| a | 亞 | .458 | b | 布 | .700 | ye | 耶 | .667 |
| | 阿 | .271 | | λ | .133 | | 葉 | .333 |
| | 艾 | .059 | | 伯 | .033 | z | 茲 | .476 |
| | λ | .051 | | 柏 | .033 | | λ | .286 |
| an | 安 | .923 | an | 安 | .923 | | 士 | .095 |
| | 恩 | .077 | | 恩 | .077 | | 芝 | .048 |

## 3.3 Finding and locating translations

At runtime, *TermMine* follows the following steps to translate a given term *E*:

(1) **Webpage retrieval**. The term *E* is submitted to a Web search engine with the language option set to the target language to obtain a set of summaries.

(2) **Matching patterns against summaries**. The surface patterns *P* learned in the training phase are applied to match *E* in the tokenized summaries, to extract a token that matches the *F* symbol in the pattern.

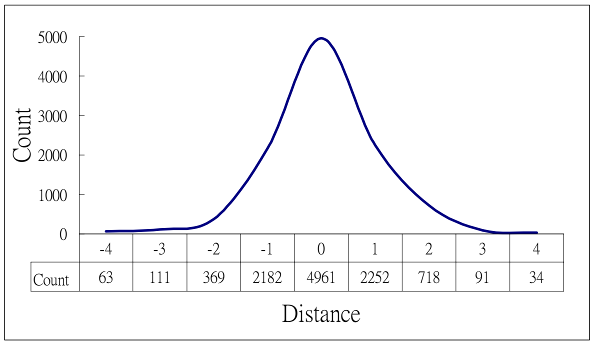
(3) **Generating candidates**. We take the distinct substrings *C* of all matched *F*s as the candidates.

(4) **Ranking candidates**. We evaluate and select translation candidates by using both data redundancy and the transliteration model. Candidates with a count or transliteration probability lower than empirically determined thresholds are discarded.

   I. **Data redundancy.** We rank translation candidates by numbers of instances it appeared in the retrieved summaries.

   II. **Transliteration Model.** For upper-case *E*, we assume *E* is a proper name and evaluate each candidate translation *C* by the likelihood of *C* as the transliteration of *E* using the transliteration model described in (Lin, Wu and Chang 2004).

Figure 4. The distribution of distances between source and target terms in Web pages.



| | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| Count | 63 | 111 | 369 | 2182 | 4961 | 2252 | 718 | 91 | 34 |

Figure 5. The distribution of distances between source and target terms in Web pages.

| Pattern | Count | Acc. Percent | Example | distance |
|---|---|---|---|---|
| FE | 3036 | 28.1% | 亞特拉斯 ATLAS | 0 |
| EF | 1925 | 45.9% | Elton John 艾爾頓強 | 0 |
| E(F | 1485 | 59.7% | Austria(奧地利 | -1 |
| F（E | 1251 | 71.2% | 亞特拉斯（Atlas | 1 |
| F(E | 361 | 74.6% | 亞特拉斯(Atlas | 1 |
| F.E | 203 | 76.5% | Peter Pan. 小飛俠 | 1 |
| EwF | 197 | 78.3% | 加州 Northern California | -1 |
| E,F | 153 | 79.7% | Mexico, 墨西哥 | -1 |
| F》（E | 137 | 81.0% | 鐵達尼號》（Titanic | 2 |
| F」（E | 119 | 82.1% | 亞特拉斯」（Atlas | 2 |

(5) **Expanding the tentative translation**. Based on a heuristics proposed by Smadja (1991) to expand bigrams to full collocations, we extend the top-ranking candidate with count *n* on both sides, while keeping the count greater than *n*/2 (empirically determined). Note that the constant *n* is set to 10 in the experiment described in Section 4.

(6) Final ranking. Rank the expanded versions of candidates by occurrence count and output the ranked list.

## 4 Experimental results

We took the answers of the first 215 questions on a quiz Website (www.quiz-zone.co.uk) and hand-translations as the training data to obtain a of surface patterns. For all but 17 source terms, we are able to find at least 3 instances of co-occurring of source term and translation. Figure 4 shows distribution of the distances between co-occurring source and target terms. The distances tend to concentrate between - 3 and + 3 (10,680 out of 12,398 instances, or 86%). The 212 surface patterns obtained from these 10,860 instances, have a very skew distribution with the ten most frequent surface patterns accounting for 82% of the cases (see Figure 5). In addition to source-target surface patterns, we also trained a transliteration model (see Figure 3) on 1,800 bilingual proper names appearing in Taiwanese editions of *Scientific American* magazine.

Test results on a set of 300 randomly selected proper names and technical terms from Encyclopedia Britannica indicate that *TermMine* produces 300 top-ranking answers, of which 263 is the exact translations (86%) and 293 contain the answer key (98%). In comparison, the online machine translation service, *Google translate* produces only 156 translations in full, with 103 (34%) matching the answer key exactly, and 145 (48%) containing the answer key.

## 5 Conclusion

We present a novel Web-based, data-intensive approach to terminology translation from English to Mandarin Chinese. Experimental results and contrastive evaluation indicate significant improvement over previous work and a state-of-sate commercial MT system.

## References

Y. Cao and H. Li. (2002). *Base Noun Phrase Translation Using Web Data and the EM Algorithm*, In Proc. of COLING 2002, pp.127-133.

W. Hutchins and H. Somers. (1992). *An Introduction to Machine Translation*. Academic Press.

K. Knight, J. Graehl. (1998). *Machine Transliteration*. In Journal of Computational Linguistics 24(4), pp.599-612.

P. Koehn, K. Knight. (2003). *Feature-Rich Statistical Translation of Noun Phrases*. In Proc. of ACL 2003, pp.311-318.

K. L. Kwok, *The Chinet system*. (2004). (personal communication).

T. Lin, J.C. Wu, J. S. Chang. (2004). *Extraction of Name and Transliteration in Monolingual and Parallel Corpora*. In Proc. of AMTA 2004, pp.177-186.

M. Nagata, T. Saito, and K. Suzuki. (2001). *Using the Web as a bilingual dictionary*. In Proc. of ACL 2001 DD-MT Workshop, pp.95-102.

F. A. Smadja. (1991). *From N-Grams to Collocations: An Evaluation of Xtract*. In Proc. of ACL 1991, pp.279-284.

# SPEECH OGLE: Indexing Uncertainty for Spoken Document Search

**Ciprian Chelba** and **Alex Acero**
Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
{chelba, alexac}@microsoft.com

## Abstract

The paper presents the Position Specific Posterior Lattice (PSPL), a novel lossy representation of automatic speech recognition lattices that naturally lends itself to efficient indexing and subsequent relevance ranking of spoken documents.

In experiments performed on a collection of lecture recordings — MIT iCampus data — the spoken document ranking accuracy was improved by 20% relative over the commonly used baseline of indexing the 1-best output from an automatic speech recognizer.

The inverted index built from PSPL lattices is compact — about 20% of the size of 3-gram ASR lattices and 3% of the size of the uncompressed speech — and it allows for extremely fast retrieval. Furthermore, little degradation in performance is observed when pruning PSPL lattices, resulting in even smaller indexes — 5% of the size of 3-gram ASR lattices.

## 1 Introduction

Ever increasing computing power and connectivity bandwidth together with falling storage costs result in an overwhelming amount of data of various types being produced, exchanged, and stored. Consequently, search has emerged as a key application as more and more data is being saved (Church, 2003). Text search in particular is the most active area, with applications that range from web and private network search to searching for private information residing on one's hard-drive.

Speech search has not received much attention due to the fact that large collections of untranscribed spoken material have not been available, mostly due to storage constraints. As storage is becoming cheaper, the availability and usefulness of large collections of spoken documents is limited strictly by the lack of adequate technology to exploit them.

Manually transcribing speech is expensive and sometimes outright impossible due to privacy concerns. This leads us to exploring an automatic approach to searching and navigating spoken document collections (Chelba and Acero, 2005).

## 2 Text Document Retrieval in the Early Google Approach

Aside from the use of PageRank for relevance ranking, the early Google also uses both *proximity* and *context* information heavily when assigning a relevance score to a given document (Brin and Page, 1998), Section 4.5.1.

For each given query term $q_i$ one retrieves the list of *hits* corresponding to $q_i$ in document $D$. Hits can be of various types depending on the *context* in which the hit occurred: title, anchor text, etc. Each type of hit has its own *type-weight* and the type-weights are indexed by type.

For a single word query, their ranking algorithm takes the inner-product between the type-weight vector and a vector consisting of count-weights (tapered counts such that the effect of large counts is discounted) and combines the resulting score with

PageRank in a final relevance score.

For multiple word queries, terms co-occurring in a given document are considered as forming different *proximity-types* based on their proximity, from adjacent to “not even close”. Each proximity type comes with a proximity-weight and the relevance score includes the contribution of proximity information by taking the inner product over all types, including the proximity ones.

## 3 Position Specific Posterior Lattices

As highlighted in the previous section, position information is crucial for being able to evaluate proximity information when assigning a relevance score to a given document.

In the spoken document case however, we are faced with a dilemma. On one hand, using 1-best ASR output as the transcription to be indexed is suboptimal due to the high WER, which is likely to lead to low recall — query terms that were in fact spoken are wrongly recognized and thus not retrieved. On the other hand, ASR lattices do have a much better WER — in our case the 1-best WER was 55% whereas the lattice WER was 30% — but the position information is not readily available.

The occurrence of a given word in a lattice obtained from a given spoken document is uncertain and so is the position at which the word occurs in the document. However, the ASR lattices do contain the information needed to evaluate proximity information, since on a given path through the lattice we can easily assign a position index to each link/word in the normal way. Each path occurs with a given posterior probability, easily computable from the lattice, so in principle one could index *soft-hits* which specify *(document id, position, posterior probability)* for each word in the lattice.

A simple dynamic programming algorithm which is a variation on the standard forward-backward algorithm can be employed for performing this computation. The computation for the backward probability $\beta_n$ stays unchanged (Rabiner, 1989) whereas during the forward pass one needs to split the forward probability arriving at a given node $n$, $\alpha_n$, according to the length of the partial paths that start at the start node of the lattice and end at node $n$:

$$\alpha_n[l] = \sum_{\pi:end(\pi)=n, length(\pi)=l} P(\pi)$$

The posterior probability that a given node $n$ occurs at position $l$ is thus calculated using:

$$P(n,l|LAT) = \frac{\alpha_n[l] \cdot \beta_n}{norm(LAT)}$$

The posterior probability of a given word $w$ occurring at a given position $l$ can be easily calculated using:

$$P(w,l|LAT) = \sum_{n \ s.t. \ P(n,l)>0} P(n,l|LAT) \cdot \delta(w, word(n))$$

The Position Specific Posterior Lattice (PSPL) is nothing but a representation of the $P(w,l|LAT)$ distribution. For details on the algorithm and properties of PSPL please see (Chelba and Acero, 2005).

## 4 Spoken Document Indexing and Search Using PSPL

Speech content can be very long. In our case the speech content of a typical spoken document was approximately 1 hr long. It is customary to segment a given speech file in shorter segments. A spoken document thus consists of an ordered list of segments. For each segment we generate a corresponding PSPL lattice. Each document and each segment in a given collection are mapped to an integer value using a *collection descriptor file* which lists all documents and segments.

The soft hits for a given word are stored as a vector of entries sorted by `(document id, segment id)`. Document and segment boundaries in this array, respectively, are stored separately in a map for convenience of use and memory efficiency. The *soft index* simply lists all hits for every word in the ASR vocabulary; each word entry can be stored in a separate file if we wish to augment the index easily as new documents are added to the collection.

### 4.1 Speech Content Relevance Ranking Using PSPL Representation

Consider a given query $\mathcal{Q} = q_1 \ldots q_i \ldots q_Q$ and a spoken document $D$ represented as a PSPL. Our ranking scheme follows the description in Section 2.

For all query terms, a 1-gram score is calculated by summing the PSPL posterior probability across all segments $s$ and positions $k$. This is equivalent to calculating the expected count of a given query term $q_i$ according to the PSPL probability distribution $P(w_k(s)|D)$ for each segment $s$ of document $D$. The results are aggregated in a common value $S_{1-gram}(D, \mathcal{Q})$:

$$S(D, q_i) = \log\left[1 + \sum_s \sum_k P(w_k(s) = q_i|D)\right]$$
$$S_{1-gram}(D, \mathcal{Q}) = \sum_{i=1}^{Q} S(D, q_i) \quad (1)$$

Similar to (Brin and Page, 1998), the logarithmic tapering off is used for discounting the effect of large counts in a given document.

Our current ranking scheme takes into account proximity in the form of matching $N$-grams present in the query. Similar to the 1-gram case, we calculate an expected tapered-count for each N-gram $q_i \ldots q_{i+N-1}$ in the query and then aggregate the results in a common value $S_{N-gram}(D, \mathcal{Q})$ for each order $N$:

$$S(D, q_i \ldots q_{i+N-1}) = \log\left[1 + \textstyle\sum_s \sum_k \prod_{l=0}^{N-1} P(w_{k+l}(s) = q_{i+l}|D)\right]$$
$$S_{N-gram}(D, \mathcal{Q}) = \sum_{i=1}^{Q-N+1} S(D, q_i \ldots q_{i+N-1}) \quad (2)$$

The different proximity types, one for each $N$-gram order allowed by the query length, are combined by taking the inner product with a vector of weights.

$$S(D, \mathcal{Q}) = \sum_{N=1}^{Q} w_N \cdot S_{N-gram}(D, \mathcal{Q})$$

It is worth noting that the transcription for any given segment can also be represented as a PSPL with exactly one word per position bin. It is easy to see that in this case the relevance scores calculated according to Eq. (1-2) are the ones specified by 2.

Only documents containing all the terms in the query are returned. We have also enriched the query language with the "quoted functionality" that allows us to retrieve only documents that contain exact PSPL matches for the quoted phrases, e.g. the query `''L M'' tools` will return only documents containing occurrences of `L M` and of `tools`.

## 5 Experiments

We have carried all our experiments on the iCampus corpus (Glass et al., 2004) prepared by MIT CSAIL. The main advantages of the corpus are: realistic speech recording conditions — all lectures are recorded using a lapel microphone — and the availability of accurate manual transcriptions — which enables the evaluation of a SDR system against its text counterpart.

The corpus consists of about 169 hours of lecture materials. Each lecture comes with a word-level manual transcription that segments the text into semantic units that could be thought of as sentences; word-level time-alignments between the transcription and the speech are also provided. The speech was segmented at the sentence level based on the time alignments; each lecture is considered to be a spoken document consisting of a set of one-sentence long segments determined this way. The final collection consists of 169 documents, 66,102 segments and an average document length of 391 segments.

### 5.1 Spoken Document Retrieval

Our aim is to narrow the gap between speech and text document retrieval. We have thus taken as our reference the output of a standard retrieval engine working according to one of the TF-IDF flavors. The engine indexes the manual transcription using an unlimited vocabulary. All retrieval results presented in this section have used the standard `trec_eval` package used by the TREC evaluations.

The PSPL lattices for each segment in the spoken document collection were indexed. In terms of relative size on disk, the uncompressed speech for the first 20 lectures uses 2.5GB, the ASR 3-gram lattices use 322MB, and the corresponding index derived from the PSPL lattices uses 61MB.

In addition, we generated the PSPL representation of the manual transcript and of the 1-best ASR output and indexed those as well. This allows us to compare our retrieval results against the results obtained using the reference engine when working on the same text document collection.

### 5.1.1 Query Collection and Retrieval Setup

We have asked a few colleagues to issue queries against a demo shell using the index built from the manual transcription.We have collected 116 queries in this manner. The query out-of-vocabulary rate (Q-OOV) was 5.2% and the average query length was 1.97 words. Since our approach so far does not index sub-word units, we cannot deal with OOV query words. We have thus removed the queries which contained OOV words — resulting in a set of 96 queries.

### 5.1.2 Retrieval Experiments

We have carried out retrieval experiments in the above setup. Indexes have been built from: `trans`, manual transcription filtered through ASR vocabulary; `1-best`, ASR 1-best output; `lat`, PSPL lattices. Table 1 presents the results. As a sanity check,

| | `trans` | `1-best` | `lat` |
|---|---|---|---|
| # docs retrieved | 1411 | 3206 | 4971 |
| # relevant docs | 1416 | 1416 | 1416 |
| # rel retrieved | 1411 | 1088 | 1301 |
| MAP | 0.99 | 0.53 | 0.62 |
| R-precision | 0.99 | 0.53 | 0.58 |

Table 1: Retrieval performance on indexes built from transcript, ASR 1-best and PSPL lattices

the retrieval results on transcription — `trans` — match almost perfectly the reference. The small difference comes from stemming rules that the baseline engine is using for query enhancement which are not replicated in our retrieval engine.

The results on lattices (`lat`) improve significantly on (`1-best`) — 20% relative improvement in mean average precision (MAP). Table 2 shows the retrieval accuracy results as well as the index size for various pruning thresholds applied to the `lat` PSPL. MAP performance increases with PSPL depth, as expected. A good compromise between accuracy and index size is obtained for a pruning threshold of 2.0: at very little loss in MAP one could use an index that is only 20% of the full index.

## 6 Conclusions and Future work

We have developed a new representation for ASR lattices — the Position Specific Posterior Lattice — that lends itself to indexing speech content. The retrieval results obtained by indexing the PSPL are 20% better than when using the ASR 1-best output.

The techniques presented can be applied to indexing contents of documents when uncertainty is present: optical character recognition, handwriting recognition are examples of such situations.

| pruning threshold | MAP | R-precision | Index Size (MB) |
|---|---|---|---|
| 0.0 | 0.53 | 0.54 | 16 |
| 0.1 | 0.54 | 0.55 | 21 |
| 0.2 | 0.55 | 0.56 | 26 |
| 0.5 | 0.56 | 0.57 | 40 |
| 1.0 | 0.58 | 0.58 | 62 |
| 2.0 | <u>0.61</u> | 0.59 | <u>110</u> |
| 5.0 | 0.62 | 0.57 | 300 |
| 10.0 | 0.62 | 0.57 | 460 |
| 1000000 | 0.62 | 0.57 | 540 |

Table 2: Retrieval performance on indexes built from pruned PSPL lattices, along with index size

## 7 Acknowledgments

We would like to thank Jim Glass and T J Hazen at MIT for providing the iCampus data. We would also like to thank Frank Seide for offering valuable suggestions on our work.

## References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

Ciprian Chelba and Alex Acero. 2005. Position specific posterior lattices for indexing speech. In *Proceedings of ACL*, Ann Arbor, Michigan, June.

Kenneth Ward Church. 2003. Speech and language processing: Where have we been and where are we going? In *Proceedings of Eurospeech*, Geneva, Switzerland.

James Glass, Timothy J. Hazen, Lee Hetherington, and Chao Wang. 2004. Analysis and processing of lecture audio data: Preliminary investigations. In *HLT-NAACL 2004 Workshop: Interdisciplinary Approaches to Speech Indexing and Retrieval*, pages 9–12, Boston, Massachusetts, USA, May 6.

L. R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings IEEE*, volume 77(2), pages 257–285.

# Multimodal Generation in the COMIC Dialogue System

**Mary Ellen Foster** and **Michael White**
Institute for Communicating and Collaborative Systems
School of Informatics, University of Edinburgh
{M.E.Foster,Michael.White}@ed.ac.uk

**Andrea Setzer** and **Roberta Catizone**
Natural Language Processing Group
Department of Computer Science, University of Sheffield
{A.Setzer,R.Catizone}@dcs.shef.ac.uk

**Abstract**

We describe how context-sensitive, user-tailored output is specified and produced in the COMIC multimodal dialogue system. At the conference, we will demonstrate the user-adapted features of the dialogue manager and text planner.

## 1 Introduction

COMIC[1] is an EU IST 5th Framework project combining fundamental research on human-human interaction with advanced technology development for multimodal conversational systems. The project demonstrator system adds a dialogue interface to a CAD-like application used in bathroom sales situations to help clients redesign their rooms. The input to the system includes speech, handwriting, and pen gestures; the output combines synthesised speech, a talking head, and control of the underlying application. Figure 1 shows screen shots of the COMIC interface.

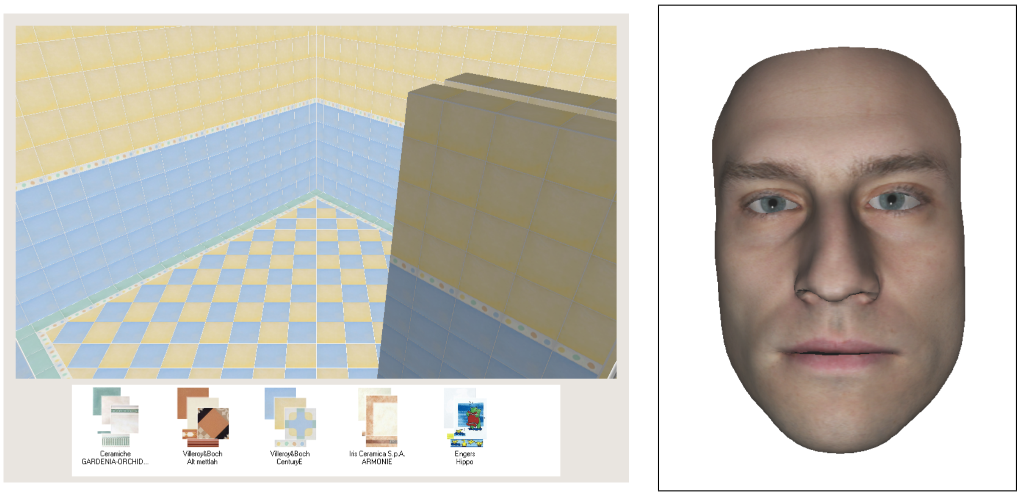
There are four main phases in the demonstrator. First, the user specifies the shape of their own bathroom, using a combination of speech input, pen-gesture recognition and handwriting recognition. Next, the user chooses a layout for the sanitary ware in the room. After that, the system guides the user in browsing through a range of tiling options for the bathroom. Finally, the user is given a three-dimensional walkthrough of the finished bathroom. We will focus on how context-sensitive, user-tailored output is generated in the third, guided-browsing phase of the interaction. Figure 2 shows a typical user request and response from COMIC in this phase. The pitch accents and multimodal actions are indicated; there is also facial emphasis corresponding to the accented words.

The primary goal of COMIC's guided-browsing phase is to help users become better informed about the range of tiling options for their bathroom. In this regard, it is similar to the web-based system M-PIRO (Isard et al., 2003), which generates personalised descriptions of museum objects, and contrasts with task-oriented embodied dialogue systems such as SmartKom (Wahlster, 2003). Since guided browsing requires extended descriptions, in COMIC we have placed greater emphasis on producing high-quality adaptive output than have previous embodied dialogue projects such as August (Gustafson et al., 1999) and Rea (Cassell et al., 1999). To generate its adaptive output, COMIC uses information from the dialogue history and the user model throughout the generation process, as in FLIGHTS (Moore et al., 2004); both systems build upon earlier work on adaptive content planning (Carenini, 2000; Walker et al., 2002). An experimental study (Foster and White, 2005) has shown that this adaptation is perceptible to users of COMIC.

## 2 Dialogue Management

The task of the Dialogue and Action Manager (DAM) is to decide what the system will show and say in response to user input. The input to the

[1]COnversational Multimodal Interaction with Computers; http://www.hcrc.ed.ac.uk/comic/.


Ceramiche GARDENIA-ORCHID...
Villeroy&Boch Alt mettlah
Villeroy&Boch CenturyE
Iris Ceramica S.p.A. ARMONIE
Engers Hippo

(a) Bathroom-design application (b) Talking head

Figure 1: Components of the COMIC interface

| | |
|---|---|
| **User** | Tell me about this design *[click on Alt Mettlach]* |
| **COMIC** | *[Look at screen]* |
| | THIS DESIGN is in the CLASSIC style. *[circle tiles]* |
| | As you can see, the colours are DARK RED and OFF WHITE. *[point at tiles]* |
| | The tiles are from the ALT METTLACH collection by VILLEROY AND BOCH. *[point at design name]* |

Figure 2: Sample COMIC input and output

DAM consists of multiple scored hypotheses containing high-level, modality-independent specifications of the user input; the output is a similar high-level specification of the system action. The DAM itself is modality-independent. For example, the input in Figure 2 could equally well have been the user simply pointing to a design on the screen, with no speech at all. This would have resulted in the same abstract DAM input, and thus in the same output: a request to show and describe the given design.

The COMIC DAM (Catizone et al., 2003) is a general-purpose dialogue manager which can handle different dialogue management styles such as system-driven, user-driven or mixed-initiative. The general-purpose part of the DAM is a simple stack architecture with a control structure; all the application-dependent information is stored in a variation of Augmented Transition Networks (ATNs) called *Dialogue Action Forms* (DAFs). These DAFs represent general dialogue moves, as well as sub-tasks or topics, and are pushed onto and popped off of the stack as the dialogue proceeds.

When processing a user input, the control structure decides whether the DAM can stay within the current topic (and thus the current DAF), or whether a topic shift has occurred. In the latter case, a new DAF is pushed onto the stack and executed. After that topic has been exhausted, the DAM returns to the previous topic automatically. The same principle holds for error handling, which is implemented at different levels in our approach.

In the guided-browsing phase of the COMIC system, the user may browse tiling designs by colour, style or manufacturer, look at designs in detail, or change the amount of border and decoration tiles. The DAM uses the system ontology to retrieve designs according to the chosen feature, and consults the user model and dialogue history to narrow down the resulting designs to a small set to be shown and described to the user.

## 3 Presentation Planning

The COMIC fission module processes high-level system-output specifications generated by the DAM. For the example in Figure 2, the DAM output indicates that the given tile design should be shown and described, and that the description must mention the style. The fission module fleshes out such specifications by selecting and structuring content, planning the surface form of the text to realise that content, choosing multimodal behaviours to accompany the text, and controlling the output of the whole schedule. In this section, we describe the planning process; output coordination is dealt with in Section 6. Full technical details of the fission module are given in (Foster, 2005).

To create the textual content of a description, the fission module proceeds as follows. First, it gathers all of the properties of the specified design from the system ontology. Next, it selects the properties to include in the description, using information from the dialogue history and the user model, along with any properties specifically requested by the dialogue manager. It then creates a structure for the selected properties and creates logical forms as input for the OpenCCG surface realiser. The logical forms may include explicit alternatives in cases where there are multiple ways of expressing a property; for example, it could say either *This design is in the classic style* or *This design is classic*. OpenCCG makes use of statistical language models to choose among such alternatives. This process is described in detail in (Foster and White, 2004; Foster and White, 2005).

In addition to text, the output of COMIC also incorporates multimodal behaviours including prosodic specifications for the speech synthesiser (pitch accents and boundary tones), facial behaviour specifications (expressions and gaze shifts), and deictic gestures at objects on the application screen using a simulated pointer. Pitch accents and boundary tones are selected by the realiser based on the context-sensitive information-structure annotations (theme/rheme; marked/unmarked) included in the logical forms. At the moment, the other multimodal coarticulations are specified directly by the fission module, but we are currently experimenting with using the OpenCCG realiser's language models to choose them, using example-driven techniques.

## 4 Surface Realisation

Surface realisation in COMIC is performed by the OpenCCG[2] realiser, a practical, open-source realiser based on Combinatory Categorial Grammar (CCG) (Steedman, 2000b). It employs a novel ensemble of methods for improving the efficiency of CCG realisation, and in particular, makes integrated use of *n*-gram scoring of possible realisations in its chart realisation algorithm (White, 2004; White, 2005). The *n*-gram scoring allows the realiser to work in "anytime" mode—able at any time to return the highest-scoring complete realisation—and ensures that a good realisation can be found reasonably quickly even when the number of possibilities is exponential. This makes it particularly suited for use in an interactive dialogue system such as COMIC.

In COMIC, the OpenCCG realiser uses factored language models (Bilmes and Kirchhoff, 2003) over words and multimodal coarticulations to select the highest-scoring realisation licensed by the grammar that satisfies the specification given by the fission module. Steedman's (Steedman, 2000a) theory of information structure and intonation is used to constrain the choice of pitch accents and boundary tones for the speech synthesiser.

## 5 Speech Synthesis

The COMIC speech-synthesis module is implemented as a client to the Festival speech-synthesis system.[3] We take advantage of recent advances in version 2 of Festival (Clark et al., 2004) by using a custom-built unit-selection voice with support for APML prosodic annotation (de Carolis et al., 2004). Experiments have shown that synthesised speech with contextually appropriate prosodic features can be perceptibly more natural (Baker et al., 2004).

Because the fission module needs the timing information from the speech synthesiser to finalise the schedules for the other modalities, the synthesiser first prepares and stores the waveform for its input text; the sound is then played at a later time, when the fission module indicates that it is required.

[2]`http://openccg.sourceforge.net/`
[3]`http://www.cstr.ed.ac.uk/projects/festival/`

## 6 Output Coordination

In addition to planning the presentation content as described earlier, the fission module also controls the system output to ensure that all parts of the presentation are properly coordinated, using the timing information returned by the speech synthesiser to create a full schedule for the turn to be generated.

As described in (Foster, 2005), the fission module allows multiple segments to be prepared in advance, even while the preceding segments are being played. This serves to minimise the output delay, as there is no need to wait until a whole turn is fully prepared before output begins, and the time taken to speak the earlier parts of the turn can also be used to prepare the later parts.

## 7 Acknowledgements

This work was supported by the COMIC project (IST-2001-32311). This paper describes only part of the work done in the project; please see `http://www.hcrc.ed.ac.uk/comic/` for full details. We thank the other members of COMIC for their collaboration during the course of the project.

## References

Rachel Baker, Robert A.J. Clark, and Michael White. 2004. Synthesizing contextually appropriate intonation in limited domains. In *Proceedings of 5th ISCA workshop on speech synthesis*.

Jeff Bilmes and Katrin Kirchhoff. 2003. Factored language models and general parallelized backoff. In *Proceedings of HLT-03*.

Giuseppe Carenini. 2000. *Generating and Evaluating Evaluative Arguments*. Ph.D. thesis, Intelligent Systems Program, University of Pittsburgh.

Justine Cassell, Timothy Bickmore, Mark Billinghurst, Lee Campbell, Kenny Chang, Hannes Vilhjálmsson, and Hao Yan. 1999. Embodiment in conversational interfaces: Rea. In *Proceedings of CHI99*.

Roberta Catizone, Andrea Setzer, and Yorick Wilks. 2003. Multimodal dialogue management in the COMIC project. In *Proceedings of EACL 2003 Workshop on Dialogue Systems: Interaction, adaptation, and styles of management*.

Robert A.J. Clark, Korin Richmond, and Simon King. 2004. Festival 2 – build your own general purpose unit selection speech synthesiser. In *Proceedings of 5th ISCA workshop on speech synthesis*.

Berardina de Carolis, Catherine Pelachaud, Isabella Poggi, and Mark Steedman. 2004. APML, a mark-up language for believable behaviour generation. In H Prendinger, editor, *Life-like Characters, Tools, Affective Functions and Applications*, pages 65–85. Springer.

Mary Ellen Foster and Michael White. 2004. Techniques for text planning with XSLT. In *Proceedings of NLPXML-2004*.

Mary Ellen Foster and Michael White. 2005. Assessing the impact of adaptive generation in the COMIC multimodal dialogue system. In *Proceedings of IJCAI-2005 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*. To appear.

Mary Ellen Foster. 2005. Interleaved planning and output in the COMIC fission module. Submitted.

Joakim Gustafson, Nikolaj Lindberg, and Magnus Lundeberg. 1999. The August spoken dialogue system. In *Proceedings of Eurospeech 1999*.

Amy Isard, Jon Oberlander, Ion Androtsopoulos, and Colin Matheson. 2003. Speaking the users' languages. *IEEE Intelligent Systems*, 18(1):40–45.

Johanna Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. 2004. Generating tailored, comparative descriptions in spoken dialogue. In *Proceedings of FLAIRS 2004*.

Mark Steedman. 2000a. Information structure and the syntax-phonology interface. *Linguistic Inquiry*, 31(4):649–689.

Mark Steedman. 2000b. *The Syntactic Process*. MIT Press.

Wolfgang Wahlster. 2003. SmartKom: Symmetric multimodality in an adaptive and reusable dialogue shell. In *Proceedings of the Human Computer Interaction Status Conference 2003*.

M.A. Walker, S. Whittaker, A. Stent, P. Maloor, J.D. Moore, M. Johnston, and G. Vasireddy. 2002. Speech-plans: Generating evaluative responses in spoken dialogue. In *Proceedings of INLG 2002*.

Michael White. 2004. Reining in CCG chart realization. In *Proceedings of INLG 2004*.

Michael White. 2005. Efficient realization of coordinate structures in Combinatory Categorial Grammar. *Research on Language and Computation*. To appear.

# Language Independent Extractive Summarization

**Rada Mihalcea**
Department of Computer Science and Engineering
University of North Texas
rada@cs.unt.edu

## Abstract

We demonstrate *TextRank* – a system for unsupervised extractive summarization that relies on the application of iterative graph-based ranking algorithms to graphs encoding the cohesive structure of a text. An important characteristic of the system is that it does not rely on any language-specific knowledge resources or any manually constructed training data, and thus it is highly portable to new languages or domains.

## 1 Introduction

Given the overwhelming amount of information available today, on the Web and elsewhere, techniques for efficient automatic text summarization are essential to improve the access to such information. Algorithms for extractive summarization are typically based on techniques for sentence extraction, and attempt to identify the set of sentences that are most important for the understanding of a given document. Some of the most successful approaches to extractive summarization consist of supervised algorithms that attempt to learn what makes a good summary by training on collections of summaries built for a relatively large number of training documents, e.g. (Hirao et al., 2002), (Teufel and Moens, 1997). However, the price paid for the high performance of such supervised algorithms is their inability to easily adapt to new languages or domains, as new training data are required for each new type of data. *TextRank* (Mihalcea and Tarau, 2004), (Mihalcea, 2004) is specifically designed to address this problem, by using an extractive summarization technique that does not require any training data or any language-specific knowledge sources. *TextRank* can be effectively applied to the summarization of documents in different languages without any modifications of the algorithm and without any requirements for additional data. Moreover, results from experiments performed on standard data sets have demonstrated that the performance of *TextRank* is competitive with that of some of the best summarization systems available today.

## 2 Extractive Summarization

Ranking algorithms, such as Kleinberg's $HITS$ algorithm (Kleinberg, 1999) or Google's $PageRank$ (Brin and Page, 1998) have been traditionally and successfully used in Web-link analysis, social networks, and more recently in text processing applications. In short, a graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. The basic idea implemented by the ranking model is that of *voting* or *recommendation*. When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex.

These graph ranking algorithms are based on a random walk model, where a walker takes random steps on the graph, with the walk being modeled as a Markov process – that is, the decision on what edge to follow is solely based on the vertex where the walker is currently located. Under certain conditions, this model converges to a stationary distribution of probabilities associated with vertices in the graph, representing the probability of finding the walker at a certain vertex in the graph. Based on the Ergodic theorem for Markov chains (Grimmett and Stirzaker, 1989), the algorithms are guaranteed to converge if the graph is both aperiodic and irreducible. The first condition is achieved for any graph that is a non-bipartite graph, while the second condition holds for any strongly connected graph. Both these conditions are achieved in the graphs constructed for the extractive summarization application implemented in *TextRank*.

While there are several graph-based ranking algorithms previously proposed in the literature, we focus on two algorithms, namely $PageRank$ (Brin and Page, 1998) and $HITS$ (Kleinberg, 1999).

Let $G = (V, E)$ be a directed graph with the set of vertices $V$ and set of edges $E$, where $E$ is a subset of $V \times V$. For a given vertex $V_i$, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex $V_i$ points to (successors).

## 2.1 PageRank

$PageRank$ (Brin and Page, 1998) is perhaps one of the most popular ranking algorithms, and was designed as a method for Web link analysis. Unlike other graph ranking algorithms, $PageRank$ integrates the impact of both incoming and outgoing links into one single model, and therefore it produces only one set of scores:

$$PR(V_i) = (1 - d) + d * \sum_{V_j \in In(V_i)} \frac{PR(V_j)}{|Out(V_j)|} \quad (1)$$

where $d$ is a parameter that is set between 0 and 1, and has the role of integrating random jumps into the random walking model.

## 2.2 HITS

$HITS$ (Hyperlinked Induced Topic Search) (Kleinberg, 1999) is an iterative algorithm that was designed for ranking Web pages according to their degree of "authority". The $HITS$ algorithm makes a distinction between "authorities" (pages with a large number of incoming links) and "hubs" (pages with a large number of outgoing links). For each vertex, $HITS$ produces two sets of scores – an "authority" score, and a "hub" score:

$$HITS_A(V_i) = \sum_{V_j \in In(V_i)} HITS_H(V_j) \quad (2)$$

$$HITS_H(V_i) = \sum_{V_j \in Out(V_i)} HITS_A(V_j) \quad (3)$$

Starting from arbitrary values assigned to each node in the graph, the ranking algorithm iterates until convergence below a given threshold is achieved. After running the algorithm, a score is associated with each vertex, which represents the *importance* of that vertex within the graph. Note that the final values are not affected by the choice of the initial value, only the number of iterations to convergence may be different.

When the graphs are built starting with natural language texts, it may be useful to integrate into the graph model the *strength* of the connection between two vertices $V_i$ and $V_j$, indicated as a weight $w_{ij}$ added to the corresponding edge. Consequently, the ranking algorithm is adapted to include edge weights, e.g. for $PageRank$ the score is determined using the following formula (a similar change can be applied to the $HITS$ algorithm):

$$PR^W(V_i) = (1-d)+d* \sum_{V_j \in In(V_i)} w_{ji} \frac{PR^W(V_j)}{\sum\limits_{V_k \in Out(V_j)} w_{kj}} \quad (4)$$

While the final vertex scores (and therefore rankings) for weighted graphs differ significantly as compared to their unweighted alternatives, the number of iterations to convergence and the shape of the convergence curves is almost identical for weighted and unweighted graphs.

For the task of single-document extractive summarization, the goal is to rank the sentences in a given text with respect to their importance for the overall understanding of the text. A graph is therefore constructed by adding a vertex for each sentence in the text, and edges between vertices are established using sentence inter-connections, defined using a simple similarity relation measured as a function of content overlap. Such a relation between two sentences can be seen as a process of *recommendation*: a sentence that addresses certain concepts in a text gives the reader a *recommendation* to refer to other sentences in the text that address the same concepts, and therefore a link can be drawn between any two such sentences that share common content.

The overlap of two sentences can be determined simply as the number of common tokens between the lexical representations of the two sentences, or it can be run through filters that e.g. eliminate stopwords, count only words of a certain category, etc. Moreover, to avoid promoting long sentences, we use a normalization factor and divide the content overlap of two sentences with the length of each sentence.

The resulting graph is highly connected, with a weight associated with each edge, indicating the strength of the connections between various sentence pairs in the text. The graph can be represented as: (a) simple *undirected* graph; (b) directed weighted graph with the orientation of edges set from a sentence to sentences that follow in the text (*directed forward*); or (c) directed weighted graph with the orientation of edges set from a sentence to previous sentences in the text (*directed backward*).

After the ranking algorithm is run on the graph, sentences are sorted in reversed order of their score, and the top ranked sentences are selected for inclusion in the summary. Figure 1 shows an example of a weighted graph built for a short sample text.

**[1]** Watching the new movie, "Imagine: John Lennon," was very painful for the late Beatle's wife, Yoko Ono.
**[2]** "The only reason why I did watch it to the end is because I'm responsible for it, even though somebody else made it," she said.
**[3]** Cassettes, film footage and other elements of the acclaimed movie were collected by Ono.
**[4]** She also took cassettes of interviews by Lennon, which were edited in such a way that he narrates the picture.
**[5]** Andrew Solt ("This Is Elvis") directed, Solt and David L. Wolper produced and Solt and Sam Egan wrote it.
**[6]** "I think this is really the definitive documentary of John Lennon's life," Ono said in an interview.

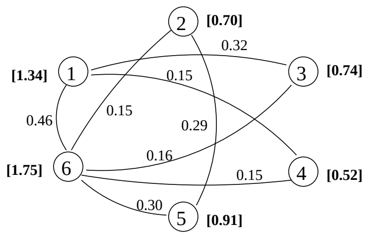


Figure 1: Graph of sentence similarities built on a sample text. Scores reflecting sentence importance are shown in brackets next to each sentence.

## 3 Evaluation

English document summarization experiments are run using the summarization test collection provided in the framework of the Document Understanding Conference (DUC). In particular, we use the data set of 567 news articles made available during the DUC 2002 evaluations (DUC, 2002), and the corresponding 100-word summaries generated for each of these documents. This is the single document summarization task undertaken by other systems participating in the DUC 2002 document summarization evaluations.

To test the language independence aspect of the algorithm, in addition to the English test collection, we also use a Brazilian Portuguese data set consisting of 100 news articles and their corresponding manually produced summaries. We use the TeMário test collection (Pardo and Rino, 2003), containing newspaper articles from online Brazilian newswire: 40 documents from *Jornal de Brasil* and 60 documents from *Folha de São Paulo*. The documents were selected to cover a variety of domains (e.g. world, politics, foreign affairs, editorials), and manual summaries were produced by an expert in Brazilian Portuguese. Unlike the summaries produced for the English DUC documents – which had a length requirement of approximately 100 words, the length of the summaries in the TeMário data set is constrained relative to the length of the corresponding documents, i.e. a summary has to account for about 25-30% of the original document. Consequently, the automatic summaries generated for the documents in this collection are not restricted to 100 words, as in the English experiments, but are required to have a length comparable to the corresponding manual summaries, to ensure a fair evaluation.

For evaluation, we are using the ROUGE evaluation toolkit[1], which is a method based on Ngram statistics, found to be highly correlated with human evaluations (Lin and Hovy, 2003). The evaluation is done using the Ngram(1,1) setting of ROUGE, which was found to have the highest correlation with human judgments, at a confidence level of 95%.

Table 2 shows the results obtained on these two data sets for different graph settings. The table also lists baseline results, obtained on summaries generated by

[1] ROUGE is available at http://www.isi.edu/˜cyl/ROUGE/.

| Algorithm | Graph | | |
|---|---|---|---|
| | Undirected | Forward | Backward |
| $HITS_A^W$ | 0.4912 | 0.4584 | **0.5023** |
| $HITS_H^W$ | 0.4912 | **0.5023** | 0.4584 |
| $PageRank^W$ | 0.4904 | 0.4202 | **0.5008** |
| Baseline | 0.4799 | | |

Table 1: English single-document summarization.

| Algorithm | Graph | | |
|---|---|---|---|
| | Undirected | Forward | Backward |
| $HITS_A^W$ | 0.4814 | 0.4834 | 0.5002 |
| $HITS_H^W$ | 0.4814 | 0.5002 | 0.4834 |
| $PageRank^W$ | 0.4939 | 0.4574 | **0.5121** |
| Baseline | 0.4963 | | |

Table 2: Portuguese single-document summarization.

taking the first sentences in each document. By ways of comparison, the best participating system in DUC 2002 was a *supervised* system that led to a ROUGE score of 0.5011.

For both data sets, *TextRank* applied on a *directed backward* graph structure exceeds the performance achieved through a simple (but powerful) baseline. These results prove that graph-based ranking algorithms, previously found successful in Web link analysis and social networks, can be turned into a state-of-the-art tool for extractive summarization when applied to graphs extracted from texts. Moreover, due to its unsupervised nature, the algorithm was also shown to be language independent, leading to similar results and similar improvements over baseline techniques when applied on documents in different languages. More extensive experimental results with the *TextRank* system are reported in (Mihalcea and Tarau, 2004), (Mihalcea, 2004).

## 4 Conclusion

Intuitively, iterative graph-based ranking algorithms work well on the task of extractive summarization because they do not only rely on the local context of a text unit (vertex), but they also take into account information recursively drawn from the entire text (graph). Through the graphs it builds on texts, a graph-based ranking algorithm identifies connections between various entities in a text, and implements the concept of *recommendation*. In the process of identifying important sentences in a text, a sentence recommends other sentences that address similar concepts as being useful for the overall understanding of the text. Sentences that are highly recommended by other sentences are likely to be more informative for the given text, and will be therefore given a higher score.

An important aspect of the graph-based extractive summarization method is that it does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, or languages.

## Acknowledgments

We are grateful to Lucia Helena Machado Rino for making available the TeMário summarization test collection and for her help with this data set.

## References

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

DUC. 2002. Document understanding conference 2002. http://www-nlpir.nist.gov/projects/duc/.

G. Grimmett and D. Stirzaker. 1989. *Probability and Random Processes*. Oxford University Press.

T. Hirao, Y. Sasaki, H. Isozaki, and E. Maeda. 2002. Ntt's text summarization system for duc-2002. In *Proceedings of the Document Understanding Conference 2002*.

J.M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

C.Y. Lin and E.H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May.

R. Mihalcea and P. Tarau. 2004. TextRank – bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain.

R. Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Lingusitics (ACL 2004) (companion volume)*, Barcelona, Spain.

T.A.S. Pardo and L.H.M. Rino. 2003. TeMario: a corpus for automatic text summarization. Technical report, NILC-TR-03-09.

S. Teufel and M. Moens. 1997. Sentence extraction as a classification task. In *ACL/EACL workshop on "Intelligent and scalable Text summarization"*, pages 58–65, Madrid, Spain.

# SenseLearner: Word Sense Disambiguation for All Words in Unrestricted Text

**Rada Mihalcea and Andras Csomai**
Department of Computer Science and Engineering
University of North Texas
rada@cs.unt.edu, ac0225@unt.edu

## Abstract

This paper describes SENSELEARNER – a minimally supervised word sense disambiguation system that attempts to disambiguate all content words in a text using WordNet senses. We evaluate the accuracy of SENSELEARNER on several standard sense-annotated data sets, and show that it compares favorably with the best results reported during the recent SENSEVAL evaluations.

## 1 Introduction

The task of word sense disambiguation consists of assigning the most appropriate meaning to a polysemous word within a given context. Applications such as machine translation, knowledge acquisition, common sense reasoning, and others, require knowledge about word meanings, and word sense disambiguation is considered essential for all these applications.

Most of the efforts in solving this problem were concentrated so far toward targeted supervised learning, where each sense tagged occurrence of a particular word is transformed into a feature vector, which is then used in an automatic learning process. The applicability of such supervised algorithms is however limited only to those few words for which sense tagged data is available, and their accuracy is strongly connected to the amount of labeled data available at hand.

Instead, methods that address all words in unrestricted text have received significantly less attention. While the performance of such methods is usually exceeded by their supervised lexical-sample alternatives, they have however the advantage of providing larger coverage.

In this paper, we present a method for solving the semantic ambiguity of all content words in a text. The algorithm can be thought of as a minimally supervised word sense disambiguation algorithm, in that it uses a relatively small data set for training purposes, and generalizes the concepts learned from the training data to disambiguate the words in the test data set. As a result, the algorithm does not need a separate classifier for each word to be disambiguated, but instead it learns global models for general word categories.

## 2 Background

For some natural language processing tasks, such as part of speech tagging or named entity recognition, regardless of the approach considered, there is a consensus on what makes a successful algorithm. Instead, no such consensus has been reached yet for the task of word sense disambiguation, and previous work has considered a range of knowledge sources, such as local collocational clues, common membership in semantically or topically related word classes, semantic density, and others.

In recent SENSEVAL-3 evaluations, the most successful approaches for all words word sense disambiguation relied on information drawn from annotated corpora. The system developed by (Decadt et al., 2004) uses two cascaded memory-based classifiers, combined with the use of a genetic algorithm for joint parameter optimization and feature selection. A separate "word expert" is learned for each ambiguous word, using a concatenated corpus of English sense-

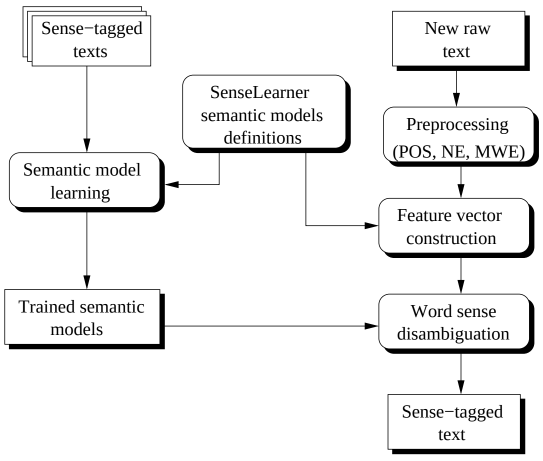

Sense−tagged texts
New raw text
SenseLearner semantic models definitions
Preprocessing (POS, NE, MWE)
Semantic model learning
Feature vector construction
Trained semantic models
Word sense disambiguation
Sense−tagged text

Figure 1: Semantic model learning in SENSELEARNER

tagged texts, including SemCor, SENSEVAL data sets, and a corpus built from WordNet examples. The performance of this system on the SENSEVAL-3 English all words data set was evaluated at 65.2%.

Another top ranked system is the one developed by (Yuret, 2004), which combines two Naive Bayes statistical models, one based on surrounding collocations and another one based on a bag of words around the target word. The statistical models are built based on SemCor and WordNet, for an overall disambiguation accuracy of 64.1%.

A different version of our own SENSELEARNER system (Mihalcea and Faruque, 2004), using three of the semantic models described in this paper, combined with semantic generalizations based on syntactic dependencies, achieved a performance of 64.6%.

## 3 SenseLearner

Our goal is to use as little annotated data as possible, and at the same time make the algorithm *general* enough to be able to disambiguate as many content words as possible in a text, and *efficient* enough so that large amounts of text can be annotated in real time. SENSELEARNER is attempting to learn general semantic models for various word categories, starting with a relatively small sense-annotated corpus. We base our experiments on SemCor (Miller et al., 1993), a balanced, semantically annotated dataset, with all content words manually tagged by trained lexicographers.

The input to the disambiguation algorithm consists of raw text. The output is a text with word meaning annotations for all open-class words.

The algorithm starts with a preprocessing stage, where the text is tokenized and annotated with part-of-speech tags; collocations are identified using a sliding window approach, where a collocation is defined as a sequence of words that forms a compound concept defined in WordNet (Miller, 1995); named entities are also identified at this stage[1].

Next, a semantic model is learned for all predefined word categories, which are defined as groups of words that share some common syntactic or semantic properties. Word categories can be of various granularities. For instance, using the SENSELEARNER learning mechanism, a model can be defined and trained to handle all the *nouns* in the test corpus. Similarly, using the same mechanism, a finer-grained model can be defined to handle all the verbs for which at least one of the meanings is of type <*move*>. Finally, small coverage models that address one word at a time, for example a model for the adjective *small*, can be also defined within the same framework. Once defined and trained, the models are used to annotate the ambiguous words in the test corpus with their corresponding meaning. Section 4 below provides details on the various models that are currently implemented in SENSELEARNER, and information on how new models can be added to the SENSELEARNER framework.

Note that the semantic models are applicable only to: (1) words that are covered by the word category defined in the models; and (2) words that appeared at least once in the training corpus. The words that are not covered by these models (typically about 10-15% of the words in the test corpus) are assigned with the most frequent sense in WordNet.

An alternative solution to this second step was suggested in (Mihalcea and Faruque, 2004), using semantic generalizations learned from dependencies identified between nodes in a conceptual network. Their approach however, although slightly more accurate, conflicted with our goal of creating an *efficient* WSD system, and therefore we opted for the simpler back-off method that employs WordNet sense frequencies.

[1]We only identify *persons*, *locations*, and *groups*, which are the named entities specifically identified in SemCor.

## 4 Semantic Models

Different semantic models can be defined and trained for the disambiguation of different word categories. Although more general than models that are built individually for each word in a test corpus (Decadt et al., 2004), the applicability of the semantic models built as part of SENSELEARNER is still limited to those words previously seen in the training corpus, and therefore their overall coverage is not 100%.

Starting with an annotated corpus consisting of all annotated files in SemCor, a separate training data set is built for each model. There are seven models provided with the current SENSELEARNER distribution, implementing the following features:

### 4.1 Noun Models

**modelNN1**: A contextual model that relies on the first noun, verb, or adjective before the target noun, and their corresponding part-of-speech tags.
**modelNNColl**: A collocation model that implements collocation-like features based on the first word to the left and the first word to the right of the target noun.

### 4.2 Verb Models

**modelVB1** A contextual model that relies on the first word before and the first word after the target verb, and their part-of-speech tags.
**modelVBColl** A collocation model that implements collocation-like features based on the first word to the left and the first word to the right of the target verb.

### 4.3 Adjective Models

**modelJJ1** A contextual model that relies on the first noun after the target adjective.
**modelJJ2** A contextual model that relies on the first word before and the first word after the target adjective, and their part-of-speech tags.
**modelJJColl** A collocation model that implements collocation-like features using the first word to the left and the first word to the right of the target adjective.

### 4.4 Defining New Models

New models can be easily defined and trained following the same SENSELEARNER learning methodology. In fact, the current distribution of SENSELEARNER includes a template for the subroutine required to define a new semantic model, which can be easily adapted to handle new word categories.

### 4.5 Applying Semantic Models

In the training stage, a feature vector is constructed for each sense-annotated word covered by a semantic model. The features are model-specific, and feature vectors are added to the training set pertaining to the corresponding model. The label of each such feature vector consists of the target word and the corresponding sense, represented as *word#sense*. Table 1 shows the number of feature vectors constructed in this learning stage for each semantic model.

To annotate new text, similar vectors are created for all content-words in the raw text. Similar to the training stage, feature vectors are created and stored separately for each semantic model.

Next, word sense predictions are made for all test examples, with a separate learning process run for each semantic model. For learning, we are using the Timbl memory based learning algorithm (Daelemans et al., 2001), which was previously found useful for the task of word sense disambiguation (Hoste et al., 2002), (Mihalcea, 2002).

Following the learning stage, each vector in the test data set is labeled with a *predicted* word and sense. If several models are simultaneously used for a given test instance, then all models have to agree in the label assigned, for a prediction to be made. If the word predicted by the learning algorithm coincides with the target word in the test feature vector, then the predicted sense is used to annotate the test instance. Otherwise, if the predicted word is different than the target word, no annotation is produced, and the word is left for annotation in a later stage.

## 5 Evaluation

The SENSELEARNER system was evaluated on the SENSEVAL-2 and SENSEVAL-3 English all words data sets, each data set consisting of three texts from the Penn Treebank corpus annotated with WordNet senses. The SENSEVAL-2 corpus includes a total of 2,473 annotated content words, and the SENSEVAL-3 corpus includes annotations for an additional set of 2,081 words. Table 1 shows precision and recall figures obtained with each semantic model on these two data sets. A baseline, computed using the most frequent sense in WordNet, is also indicated. The best results reported on these data sets are 69.0% on SENSEVAL-2 data (Mihalcea and Moldovan, 2002),

| Model | Training size | SENSEVAL-2 Precision | SENSEVAL-2 Recall | SENSEVAL-3 Precision | SENSEVAL-3 Recall |
|---|---|---|---|---|---|
| modelNN1 | 88058 | 0.6910 | 0.3257 | 0.6624 | 0.3027 |
| modelNNColl | 88058 | 0.7130 | 0.3360 | 0.6813 | 0.3113 |
| modelVB1 | 48328 | 0.4629 | 0.1037 | 0.5352 | 0.1931 |
| modelVBColl | 48328 | 0.4685 | 0.1049 | 0.5472 | 0.1975 |
| modelJJ1 | 35664 | 0.6525 | 0.1215 | 0.6648 | 0.1162 |
| modelJJ2 | 35664 | 0.6503 | 0.1211 | 0.6593 | 0.1153 |
| modelJJColl | 35664 | 0.6792 | 0.1265 | 0.6703 | 0.1172 |
| model*1/2 | 207714 | 0.6481 | 0.6481 | 0.6184 | 0.6184 |
| model*Coll | 172050 | 0.6622 | 0.6622 | 0. 6328 | 0.6328 |
| Baseline | | 63.8% | 63.8% | 60.9% | 60.9% |

Table 1: Precision and recall for the SENSELEARNER semantic models, measured on the SENSEVAL-2 and SENSEVAL-3 English all words data. Results for combinations of contextual (model*1/2) and collocational (model*Coll) models are also included.

and 65.2% on SENSEVAL-3 data (Decadt et al., 2004). Note however that both these systems rely on significantly larger training data sets, and thus the results are not directly comparable.

In addition, we also ran an experiment where a separate model was created for each individual word in the test data, with a back-off method using the most frequent sense in WordNet when no training examples were found in SEMCOR. This resulted into significantly higher complexity, with a very large number of models (about 900–1000 models for each of the SENSEVAL-2 and SENSEVAL-3 data sets), while the performance did not exceed the one obtained with the more general semantic models.

The average disambiguation precision obtained with SENSELEARNER improves significantly over the simple but competitive baseline that selects by default the “most frequent sense” from WordNet. Not surprisingly, the verbs seem to be the most difficult word class, which is most likely explained by the large number of senses defined in WordNet for this part of speech.

## 6 Conclusion

In this paper, we described and evaluated an efficient algorithm for minimally supervised word-sense disambiguation that attempts to disambiguate all content words in a text using WordNet senses. The results obtained on both SENSEVAL-2 and SENSEVAL-3 data sets are found to significantly improve over the simple but competitive baseline that chooses by default the most frequent sense, and are proved competitive with the best published results on the same data sets. SENSELEARNER is publicly available for download at *http://lit.csci.unt.edu/˜senselearner*.

## Acknowledgments

This work was partially supported by a National Science Foundation grant IIS-0336793.

## References

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. 2001. Timbl: Tilburg memory based learner, version 4.0, reference guide. Technical report, University of Antwerp.

B. Decadt, V. Hoste, W. Daelemans, and A. Van den Bosch. 2004. Gambl, genetic algorithm optimization of memory-based wsd. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

V. Hoste, W. Daelemans, I. Hendrickx, and A. van den Bosch. 2002. Evaluating the results of a memory-based word-expert approach to unrestricted word sense disambiguation. In *Proceedings of the ACL Workshop on ”Word Sense Disambiguatuion: Recent Successes and Future Directions”*, Philadelphia, July.

R. Mihalcea and E. Faruque. 2004. SenseLearner: Minimally supervised word sense disambiguation for all words in open text. In *Proceedings of ACL/SIGLEX Senseval-3*, Barcelona, Spain, July.

R. Mihalcea and D. Moldovan. 2002. Pattern learning and active feature selection for word sense disambiguation. In *Senseval 2001, ACL Workshop*, pages 127–130, Toulouse, France, July.

R. Mihalcea. 2002. Instance based learning with automatic feature selection applied to Word Sense Disambiguation. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan, August.

G. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, New Jersey.

G. Miller. 1995. Wordnet: A lexical database. *Communication of the ACM*, 38(11):39–41.

D. Yuret. 2004. Some experiments with a naive bayes wsd system. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

# Syntax-based Semi-Supervised Named Entity Tagging

**Behrang Mohit**
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260 USA
behrang@cs.pitt.edu

**Rebecca Hwa**
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260, USA
hwa@cs.pitt.edu

## Abstract

We report an empirical study on the role of syntactic features in building a semi-supervised named entity (NE) tagger. Our study addresses two questions: What types of syntactic features are suitable for extracting potential NEs to train a classifier in a semi-supervised setting? How good is the resulting NE classifier on testing instances dissimilar from its training data? Our study shows that constituency and dependency parsing constraints are both suitable features to extract NEs and train the classifier. Moreover, the classifier showed significant accuracy improvement when constituency features are combined with new dependency feature. Furthermore, the degradation in accuracy on unfamiliar test cases is low, suggesting that the trained classifier generalizes well.

## 1 Introduction

Named entity (NE) tagging is the task of recognizing and classifying phrases into one of many semantic classes such as *persons*, *organizations* and *locations*. Many successful NE tagging systems rely on a supervised learning framework where systems use large annotated training resources (Bikel et. al. 1999). These resources may not always be available for non-English domains. This paper examines the practicality of developing a syntax-based semi-supervised NE tagger. In our study we compared the effects of two types of syntactic rules (constituency and dependency) in extracting and classifying potential named entities. We train a Naive Bayes classification model on a combination of labeled and unlabeled examples with the Expectation Maximization (EM) algorithm. We find that a significant improvement in classification accuracy can be achieved when we combine both dependency and constituency extraction methods. In our experiments, we evaluate the generalization (coverage) of this bootstrapping approach under three testing schemas. Each of these schemas represented a certain level of test data coverage (recall). Although the system performs best on (unseen) test data that is extracted by the syntactic rules (i.e., similar syntactic structures as the training examples), the performance degradation is not high when the system is tested on more general test cases. Our experimental results suggest that a semi-supervised NE tagger can be successfully developed using syntax-rich features.

## 2 Previous Works and Our Approach

Supervised NE Tagging has been studied extensively over the past decade (Bikel et al. 1999, Baluja et. al. 1999, Tjong Kim Sang and De Meulder 2003). Recently, there were increasing interests in semi-supervised learning approaches. Most relevant to our study, Collins and Singer (1999) showed that a NE Classifier can be developed by bootstrapping from a small amount of labeled examples. To extract potentially useful training examples, they first parsed the sentences and looked for expressions that satisfy two constituency patterns (appositives and prepositional phrases). A small subset of these expressions was then manually labeled with their correct NE tags. The training examples were a combination of the labeled and unlabeled data. In their studies,

*Proceedings of the ACL Interactive Poster and Demonstration Sessions,*
pages 57–60, Ann Arbor, June 2005. ©2005 Association for Computational Linguistics

Collins and Singer compared several learning models using this style of semi-supervised training. Their results were encouraging, and their studies raised additional questions. First, are there other appropriate syntactic extraction patterns in addition to appositives and prepositional phrases? Second, because the test data were extracted in the same manner as the training data in their experiments, the characteristics of the test cases were biased. In this paper we examine the question of how well a semi-supervised system can classify arbitrary named entities. In our empirical study, in addition to the constituency features proposed by Collins and Singer, we introduce a new set of dependency parse features to recognize and classify NEs. We evaluated the effects of these two sets of syntactic features on the accuracy of the classification both separately and in a combined form (union of the two sets).

Figure 1 represents a general overview of our system's architecture which includes the following two levels: *NE Recognizer* and *NE Classifier.*

Section 3 and 4 describes these two levels in details and section 5 covers the results of the evaluation of our system.

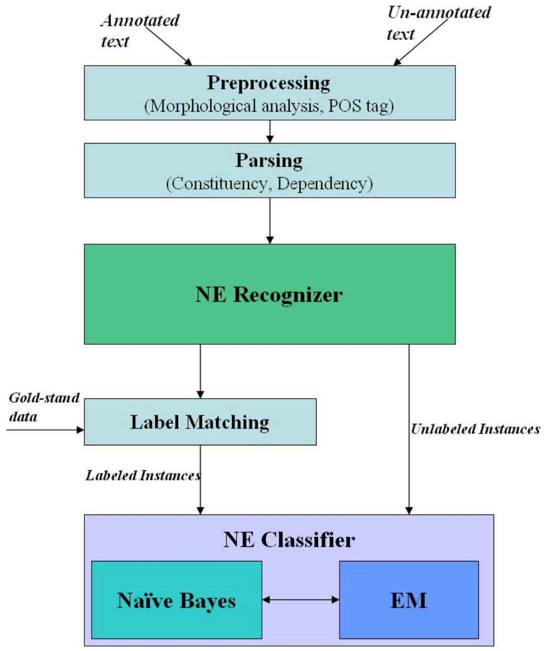


Figure 1: System's architecture

## 3 Named Entity Recognition

In this level, the system used a group of syntax-based rules to recognize and extract potential named entities from constituency and dependency parse trees. The rules are used to produce our training data; therefore they needed to have a narrow and precise coverage of each type of named entities to minimize the level of training noise.

The processing starts from construction of constituency and dependency parse trees from the input text. Potential NEs are detected and extracted based on these syntactic rules.

### 3.1 Constituency Parse Features

Replicating the study performed by Collins-Singer (1999), we used two constituency parse rules to extract a set of proper nouns (along with their associated contextual information). These two constituency rules extracted proper nouns within a noun phrase that contained an appositive phrase and a proper noun within a prepositional phrase.

### 3.2 Dependency Parse Features

We observed that a proper noun acting as the subject or the object of a sentence has a high probability of being a particular type of named entity. Thus, we expanded our syntactic analysis of the data into dependency parse of the text and extracted a set of proper nouns that act as the subjects or objects of the main verb. For each of the subjects and objects, we considered the maximum span noun phrase that included the modifiers of the subjects and objects in the dependency parse tree.

## 4 Named Entity Classification

In this level, the system assigns one of the 4 class labels *(<PER>, <ORG>, <LOC>, <NONE>)* to a given test NE. The *NONE* class is used for the expressions mistakenly extracted by syntactic features that were not a NE. We will discuss the form of the test NE in more details in section 5. The underlying model we consider is a Naïve Bayes classifier; we train it with the Expectation-Maximization algorithm, an iterative parameter estimation procedure.

### 4.1 Features

We used the following syntactic and spelling features for the classification:

*Full NE Phrase.*

*Individual word*: This binary feature indicates the presence of a certain word in the NE.

*Punctuation pattern:* The feature helps to distinguish those NEs that hold certain patterns of punctuations like *(...)* for *U.S.A.* or *(&.)* for *A&M*.
*All Capitalization:* This binary feature is mainly useful for some of the NEs that have all capital letters. such as AP, AFP, CNN, etc.
*Constituency Parse Rule:* The feature indicates which of the two constituency rule is used for extract the NE.
*Dependency Parse Rule:* The feature indicates if the NE is the subject or object of the sentence.
Except for the last two features, all features are spelling features which are extracted from the actual NE phrase. The constituency and dependency features are extracted from the NE recognition phase (section 3). Depending on the type of testing and training schema, the NEs might have 0 value for the dependency or constituency features which indicate the absence of the feature in the recognition step.

### 4.2 Naïve Bayes Classifier

We used a Naïve Bayes classifier where each NE is represented by a set of syntactic and word-level features (with various distributions) as described above. The individual words within the noun phrase are binary features. These, along with other features with multinomial distributions, fit well into Naïve Bayes assumption where each feature is dealt independently (given the class value). In order to balance the effects of the large binary features on the final class probabilities, we used some numerical methods techniques to transform some of the probabilities to the log-space.

### 4.3 Semi-supervised learning

Similar to the work of Nigam et al. (1999) on document classification, we used Expectation Maximization (EM) algorithm along with our Naïve Bayes classifier to form a semi supervised learning framework. In this framework, the small labeled dataset is used to do the initial assignments of the parameters for the Naïve Bayes classifier. After this initialization step, in each iteration the Naïve Bayes classifier classifies all of the unlabeled examples and updates its parameters based on the class probability of the unlabeled and labeled NE instances. This iterative procedure continues until the parameters reach a stable point. Subsequently the updated Naïve Bayes classifies the test instances for evaluation.

## 5 Empirical Study

Our study consists of a 9-way comparison that includes the usage of three types of training features and three types of testing schema.

### 5.1 Data

We used the data from the Automatic Content Extraction (ACE)'s entity detection track as our labeled (*gold standard*) data.[1]
For every NE that the syntactic rules extract from the input sentence, we had to find a matching NE from the gold standard data and label the extracted NE with the correct NE class label. If the extracted NE did not match any of the gold standard NEs (for the sentence), we labeled it with the <*NONE*> class label.
We also used the WSJ portion of the Penn Tree Bank as our unlabeled dataset and ran constituency and dependency analyses[2] to extract a set of unlabeled named entities for the semi-supervised classification.

### 5.2 Evaluation

In order to evaluate the effects of each group of syntactic features, we experimented with three different training strategies (using constituency rules, dependency rules or combinations of both). We conducted the comparison study with three types of test data that represent three levels of coverage (recall) for the system:
1. Gold Standard NEs: This test set contains instances taken directly from the ACE data, and are therefore independent of the syntactic rules.
2. Any single or series of proper nouns in the text: This is a heuristic for locating potential NEs so as to have the broadest coverage.
3. NEs extracted from text by the syntactic rules. This evaluation approach is similar to that of Collins and Singer. The main difference is that we have to match the extracted expressions to a pre-

[1] We only used the NE portion of the data and removed the information for other tracking and extraction tasks.
[2] We used the Collins parser (1997) to generate the constituency parse and a dependency converter (Hwa and Lopez, 2004) to obtain the dependency parse of English sentences.

labeled gold standard from ACE rather than performing manual annotations ourselves.

All tests have been performed under a 5-fold cross validation training-testing setup. Table 1 presents the accuracy of the NE classification and the size of labeled data in the different training-testing configurations. The second line of each cell shows the size of *labeled* training data and the third line shows the size of testing data. Each column presents the result for one type of the syntactic features that were used to extract NEs. Each row of the table presents one of the three testing schema. We tested the statistical significance of each of the cross-row accuracy improvements against an alpha value of 0.1 and observed significant improvement in all of the testing schemas.

| **Testing Data** | **Training Features** | | |
|---|---|---|---|
| | **Const.** | **Dep.** | **Union** |
| **Gold Standard NEs (ACE Data)** | 76.7%<br>668<br>579 | 78.5%<br>884<br>579 | 82.4%<br>1427<br>579 |
| **All Proper Nouns** | 70.2%<br>668<br>872 | 71.4%<br>884<br>872 | 76.1%<br>1427<br>872 |
| **NEs Extracted by Training Rules** | 78.2%<br>668<br>169 | 80.3%<br>884<br>217 | 85.1%<br>1427<br>354 |

Table 1: Classification Accuracy, labeled training & testing data size

Our results suggest that dependency parsing features are reasonable extraction patterns, as their accuracy rates are competitive against the model based solely on constituency rules. Moreover, they make a good complement to the constituency rules proposed by Collins and Singer, since the accuracy rates of the union is higher than either model alone. As expected, all methods perform the best when the test data are extracted in the same manner as the training examples. However, if the systems were given a well-formed named entity, the performance degradation is reasonably small, about 2% absolute difference for all training methods. The performance is somewhat lower when classifying very general test cases of all proper nouns.

## 6 Conclusion and Future Work

In this paper, we experimented with different syntactic extraction patterns and different NE recognition constraints. We find that semi-supervised methods are compatible with both constituency and dependency extraction rules. We also find that the resulting classifier is reasonably robust on test cases that are different from its training examples.

An area that might benefit from a semi-supervised NE tagger is machine translation. The semi-supervised approach is suitable for non-English languages that do not have very much annotated NE data. We are currently applying our system to Arabic. The robustness of the syntactic-based approach has allowed us to port the system to the new language with minor changes in our syntactic rules and classification features.

## Acknowledgement

We would like to thank the NLP group at Pitt and the anonymous reviewers for their valuable comments and suggestions.

## References

Shumeet Baluja, Vibhu Mittal and Rahul Sukthankar, 1999. Applying machine learning for high performance named-entity extraction. In *Proceedings of Pacific Association for Computational Linguistics.*

Daniel Bikel, Robert Schwartz & Ralph Weischedel, 1999. An algorithm that learns what's in a name. Machine Learning 34.

Michael Collins, 1997. Three generative lexicalized models for statistical parsing. *In Proceedings of the 35th Annual Meeting of the ACL.*

Michael Collins, and Yoram Singer, 1999. Unsupervised Classification of Named Entities. *In Proceedings of SIGDAT.*

A. P. Dempster, N. M. Laird and D. B. Rubin, 1977. Maximum Likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society, Series B, 39(1), 1-38.*

Rebecca Hwa and Adam Lopez, 2004. On the Conversion of Constituent Parsers to Dependency Parsers. *Technical Report TR-04-118, Department of Computer Science, University of Pittsburgh.*

Kamal Nigam, Andrew McCallum, Sebastian Thrun and Tom Mitchell, 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning 39(2/3).*

Erik F. Tjong Kim Sang and Fien De Meulder, 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *In Proceedings of CoNLL-2003.*

# Portable Translator Capable of Recognizing Characters on Signboard and Menu Captured by Built-in Camera

**Hideharu Nakajima, Yoshihiro Matsuo, Masaaki Nagata, Kuniko Saito**
NTT Cyber Space Laboratories, NTT Corporation
Yokosuka, 239-0847, Japan
{nakajima.hideharu, matsuo.yoshihiro, nagata.masaaki, saito.kuniko}
@lab.ntt.co.jp

## Abstract

We present a portable translator that recognizes and translates phrases on signboards and menus as captured by a built-in camera. This system can be used on PDAs or mobile phones and resolves the difficulty of inputting some character sets such as Japanese and Chinese if the user doesn't know their readings. Through the high speed mobile network, small images of signboards can be quickly sent to the recognition and translation server. Since the server runs state of the art recognition and translation technology and huge dictionaries, the proposed system offers more accurate character recognition and machine translation.

## 1 Introduction

Our world contains many signboards whose phrases provide useful information. These include destinations and notices in transportation facilities, names of buildings and shops, explanations at sightseeing spots, and the names and prices of dishes in restaurants. They are often written in just the mother tongue of the host country and are not always accompanied by pictures. Therefore, tourists must be provided with translations.

Electronic dictionaries might be helpful in translating words written in European characters, because key-input is easy. However, some character sets such as Japanese and Chinese are hard to input if the user doesn't know the readings such as *kana* and *pinyin*. This is a significant barrier to any translation service. Therefore, it is essential to replace keyword entry with some other input approach that supports the user when character readings are not known.

One solution is the use of optical character recognition (OCR) (Watanabe et al., 1998; Haritaoglu, 2001; Yang et al., 2002). The basic idea is the connection of OCR and machine translation (MT) (Watanabe et al., 1998) and implementation with personal data assistant (PDA) has been proposed (Haritaoglu, 2001; Yang et al., 2002). These are based on the document OCR which first tries to extract character regions; performance is weak due to the variation in lighting conditions. Although the system we propose also uses OCR, it is characterized by the use of a more robust OCR technology that doesn't first extract character regions, by language processing to offset the OCR shortcomings, and by the use of the client-server architecture and the high speed mobile network (the third generation (3G) network).

## 2 System design

**Figure 1** overviews the system architecture. After the user takes a picture by the built-in camera of a PDA, the picture is sent to a controller in a remote server. At the server side, the picture is sent to the OCR module which usually outputs many character candidates. Next, the word recognizer identifies word sequences in the candidates up to the number specified by the user. Recognized words are sent to the language translator.

The PDA is linked to the server via wireless com-

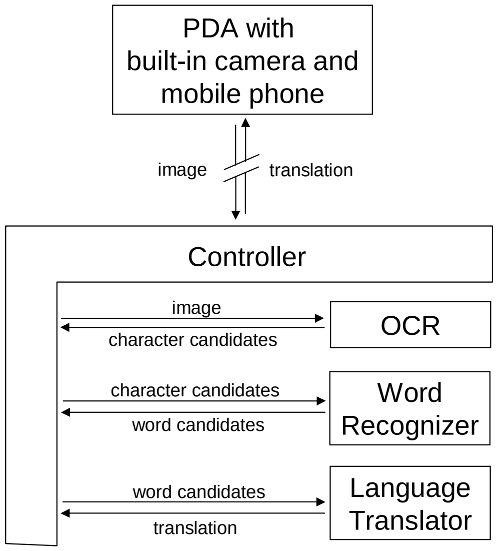


Figure 1: System architecture: http protocol is used between PDAs and the controller.

munication. The current OCR software is Windows-based while the other components are Linux programs. The PDA uses Windows.

We also implemented the system for mobile phones using the i-mode and FOMA devices provided by NTT-DoCoMo.

## 3 Each component

### 3.1 Appearance-based full search OCR

Research into the recognition of characters in natural scenes has only just begun (Watanabe et al., 1998; Haritaoglu, 2001; Yang et al., 2002; Wu et al., 2004). Many conventional approaches first extract character regions and then classify them into each character category. However, these approaches often fail at the extraction stage, because many pictures are taken under less than desirable conditions such as poor lighting, shading, strain, and distortion in the natural scene. Unless the recognition target is limited to some specific signboard (Wu et al., 2004), it is hard for the conventional OCR techniques to obtain sufficient accuracy to cover a broad range of recognition targets.

To solve this difficulty, Kusachi et al. proposed a robust character classifier (Kusachi et al., 2004). The classifier uses appearance-based character reference pattern for robust matching even under poor capture conditions, and searches the most probable

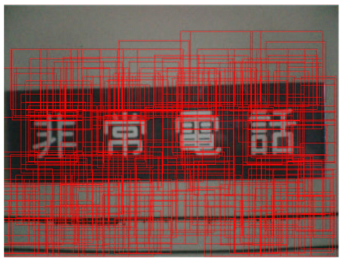


Figure 2: Many character candidates raised by appearance-based full search OCR: Rectangles denote regions of candidates. The picure shows that candidates are identified in background regions too.

region to identify candidates. As full details are given in their paper (Kusachi et al., 2004), we focus here on just its characteristic performance.

As this classifier identifies character candidates from anywhere in the picture, the precision rate is quite low, i.e. it lists a lot of wrong candidates. **Figure 2** shows a typical result of this OCR. Rectangles indicate erroneous candidates, even in background regions. On the other hand , as it identifies multiple candidates from the same location, it achieves high recall rates at each character position (over 80%) (Kusachi et al., 2004). Hence, if character positions are known, we can expect that true characters will be ranked above wrong ones, and greater word recognition accuracies would be achieved by connecting highly ranked characters in each character position. This means that location estimation becomes important.

### 3.2 Word recognition

Modern PDAs are equipped with styluses. The direct approach to obtaining character location is for the user to indicate them using the stylus. However, pointing at all the locations is tiresome, so automatic estimation is needed. Completely automatic recognition leads to extraction errors so we take the middle approach: the user specifies the beginning and ending of the character string to be recognized and translated. In **Figure 3**, circles on both ends of the string denote the user specified points. All the locations of characters along the target string are estimated from these two locations as shown in Figure 3 and all the candidates as shown in Figure 2.

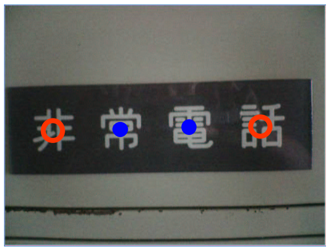


Figure 3: Two circles at the ends of the string are specified by the user with stylus. All the character locations (four locations) are automatically estimated.

### 3.2.1 Character locations

Once the user has input the end points, assumed to lie close to the centers of the end characters, the automatic location module determines the size and position of the characters in the string. Since the characters have their own regions delineated by rectangles and have x,y coordinates (as shown in Figure 2), the module considers all candidates and rates the arrangement of rectangles according to the differences in size and separation along the sequences of rectangles between both ends of the string. The sequences can be identified by any of the search algorithms used in Natural Language Processing like the forward Dynamic Programming and backward A* search (adopted in this work). The sequence with the highest score, least total difference, is selected as the true rectangle (candidate) sequence. The centers of the rectangles are taken as the locations of the characters in the string.

### 3.2.2 Word search

The character locations output by the automatic location module are not taken as specifying the correct characters, because multiple character candidates are possible at the same location. Therefore, we identify the words in the string by the probabilities of character combinations. To increase the accuracy, we consider all candidates around each estimated location and create a character matrix, an example of which is shown in **Figure 4**. At each location, we rank the candidates according to their OCR scores, the highest scores occupy the top row. Next, we apply an algorithm that consists of similar character matching, similar word retrieval, and word sequence search using language model scores

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| **非** | 吊 | **電** | 詰 |
| 乍 | **常** | 雷 | **話** |
| 匪 | 亮 | 雪 | 聞 |
| 罪 | 席 | 亀 | 諸 |
| 主 | 呈 | 巴 | 間 |
| : | : | : | : |

Figure 4: A character matrix: Character candidates are bound to each estimated location to make the matrix. Bold characters are true.

(Nagata, 1998).

The algorithm is applied from the start to the end of the string and examines all possible combinations of the characters in the matrix. At each location, the algorithm finds all words, listed in a word dictionary, that are possible given the location; that is, the first location restricts the word candidates to those that start with this character. Moreover, to counter the case in which the true character is not present in the matrix, the algorithm identifies those words in the dictionary that contain characters similar to the characters in the matrix and outputs those words as word candidates. The connectivity of neighboring words is represented by the probability defined by the language model. Finally, forward Dynamic Programming and backward A* search are used to find the word sequence with highest probability. The string in the Figure 3 is recognized as “非常電話.”

## 3.3 Language translation

Our system currently uses the ALT-J/E translation system which is a rule-based system and employs the multi-level translation method based on constructive process theory (Ikehara et al., 1991). The string in Figure 3 is translated into “Emergency telephones.”

As target language pairs will increased in future, the translation component will be replaced by statistical or corpus based translators since they offer quicker development. By using this client-server architecture on the network, we can place many task specific translation modules on server machines and flexibly select them task by task.

Table 1: Character Recognition Accuracies

| [%] | OCR | OCR+manual | OCR+auto |
|---|---|---|---|
| recall | 91 | 91 | 91 |
| precision | 12 | 82 | 80 |

## 4 Preliminary evaluation of character recognition

Because this camera base system is primarily for inputting character sets, we collected 19 pictures of signboards with a 1.2 mega pixel CCD camera for a preliminary evaluation of word recognition performance. Both ends of a string in each picture were specified on a desk-top personal computer for quick performance analysis such as tallying up the accuracy. Average string length was five characters. The language model for word recognition was basically a word bigram and trained using news paper articles.

The base OCR system returned over one hundred candidates for every picture. Though the average character recall rate was high, over 90%, wrong candidates were also numerous and the average character precision was about 12%.

The same pictures were evaluated using our method. It improved the precision to around 80% (from 12%). This almost equals the precision of about 82% obtained when the locations of all characters were manually indicated (**Table1**). Also the accuracy of character location estimation was around 95%. 11 of 19 strings (phrases) were correctly recognized.

The successfully recognized strings consisted of characters whose sizes were almost the same and they were evenly spaced. Recognition was successful even if character spacing almost equaled character size. If a flash is used to capture the image, the flash can sometimes be seen in the image which can lead to insertion error; it is recognized as a punctuation mark. However, this error is not significant since the picture taking skill of the user will improve with practice.

## 5 Conclusion and future work

Our system recognizes characters on signboards and translates them into other languages. Robust character recognition is achieved by combining high-recall and low-precision OCR and language processing.

In future, we are going to study translation qualities, prepare error-handling mechanisms for brittle OCR, MT and its combination, and explore new application areas of language computation.

**Acknowledgement**

The authors wish to thank Hisashi Ohara and Akihiro Imamura for their encouragement and Yoshinori Kusachi, Shingo Ando, Akira Suzuki, and Ken'ichi Arakawa for providing us with the use of the OCR program.

## References

Ismail Haritaoglu. 2001. InfoScope: Link from Real World to Digital Information Space. In *Proceedings of the 3rd International Conference on Ubiquitous Computing*, Springer-Verlag, pages 247-255.

Satoru Ikehara, Satoshi Shirai, Akio Yokoo and Hiromi Nakaiwa. 1991. Toward an MT System without Pre-Editing - Effects of New Methods in ALT-J/E -. In *Proceedings of the 3rd MT Summit*, pages 101-106.

Yoshinori Kusachi, Akira Suzuki, Naoki Ito, Ken'ichi Arakawa. 2004. Kanji Recognition in Scene Images without Detection of Text Fields - Robust Against Variation of Viewpoint, Contrast, and Background Texture. In *Proceedings of the 17th International Conference on Pattern Recognition*, pages 204-207.

Masaaki Nagata. 1998. Japanese OCR Error Correction using Character Shape Similarity and Statistical Language Model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 922-928.

Yasuhiko Watanabe, Yoshihiro Okada, Yeun-Bae Kim, Tetsuya Takeda. 1998. Translation Camera. In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 613–617.

Wen Wu, Xilin Chen, Jie Yang. 2004. Incremental Detection of Text on Road Signs from Video with Application to a Driving Assistant System. In *Proceedings of the ACM Multimedia 2004*, pages 852-859.

Jie Yang, Xilin Chen, Jing Zhang, Ying Zhang, Alex Waibel. 2002. Automatic Detection and Translation of Text From Natural Scenes. In *Proceedings of ICASSP*, pages 2101-2104.

# Supporting Annotation Layers for Natural Language Processing

**Preslav Nakov, Ariel Schwartz, Brian Wolf**
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
{nakov,sariel}@cs.berkeley.edu

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

## Abstract

We demonstrate a system for flexible querying against text that has been annotated with the results of NLP processing. The system supports self-overlapping and parallel layers, integration of syntactic and ontological hierarchies, flexibility in the format of returned results, and tight integration with SQL. We present a query language and its use on examples taken from the NLP literature.

## 1 Introduction

Today most natural language processing (NLP) algorithms make use of the results of previous processing steps. For example, a word sense disambiguation algorithm may combine the output of a tokenizer, a part-of-speech tagger, a phrase boundary recognizer, and a module that classifies noun phrases into semantic categories. Currently there is no standard way to represent and store the results of such processing for efficient retrieval.

We propose a framework for annotating text with the results of NLP processing and then querying against those annotations in flexible ways. The framework includes a query language and an indexing architecture for efficient retrieval, built on top of a relational database management system (RDBMS). The model allows for both hierarchical and overlapping layers of annotation as well as for querying at multiple levels of description.

In the remainder of the paper we describe related work, illustrate the annotation model and the query language and describe the indexing architecture and the experimental results, thus showing the feasibility of the approach for a variety of NLP tasks.

## 2 Related Work

There are several specialized tools for indexing and querying treebanks. (See Bird et al. (2005) for an overview and critical comparisons.) *TGrep2*[1] is a a grep-like utility for the Penn Treebank corpus of parsed Wall Street Journal texts. It allows Boolean expressions over nodes and regular expressions inside nodes. Matching uses a binary index and is performed recursively starting at the top node in the query. *TIGERSearch*[2] is associated with the German syntactic corpus TIGER. The tool is more typed than TGrep2 and allows search over discontinuous constituents that are common in German. TIGERSearch stores the corpus in a Prolog-like logical form and searches using unification matching. *LPath* is an extension of XPath with three features: immediate precedence, subtree scoping and edge alignment. The queries are executed in an SQL database (Lai and Bird, 2004). Other tree query languages include *CorpusSearch*, *Gsearch*, *Linguist's Search Engine*, *Netgraph*, *TIQL*, *VIQTORYA* etc.

Some tools go beyond the tree model and allow multiple intersecting hierarchies. *Emu* (Cassidy and Harrington, 2001) supports sequential levels of annotations over speech datasets. Hierarchical relations may exist between tokens in different levels, but precedence is defined only between elements within the same level. The queries cannot

[1]http://tedlab.mit.edu/∼dr/Tgrep2/
[2]http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/

express immediate precedence and are executed using a linear search. *NiteQL* is the query language for the MATE annotation workbench (McKelvie et al., 2001). It is highly expressive and, similarly to TIGERSearch, allows quering of intersecting hierarchies. However, the system uses XML for storage and retrieval, with an in-memory representation, which may limit its scalability.

Bird and Liberman (2001) introduce an abstract general annotation approach, based on *annotation graphs*.[3] The model is best suited for speech data, where time constraints are limited within an interval, but it is unnecessarily complex for supporting annotations on written text.

## 3 The Layered Query Language

Our framework differs from others by simultaneously supporting several key features:

- Multiple overlapping layers (which cannot be expressed in a single XML file), including self-overlapping (e.g., a word shared by two phrases from the same layer), and parallel layers, as when multiple syntactic parses span the same text.
- Integration of multiple intersecting hierarchies (e.g., MeSH, UMLS, WordNet).
- Flexible results format.
- Tight integration with SQL, including application of SQL operators over the returned results.
- Scalability to large collections such as MEDLINE (containing millions of documents).[4]

While existing systems possess some of these features, none offers all of them.

We assume that the underlying text is fairly static. While we support addition, removal and editing of annotations via a Java API, we do not optimize for efficient editing, but instead focus on compact representation, easy query formulation, easy addition and removal of layers, and straightforward translation into SQL. Below we illustrate our *Layered Query Language (LQL)* using examples from bioscience NLP.[5]

[3]http://agtk.sourceforge.net/

[4]http://www.nlm.nih.gov/pubs/factsheets/medline.html

[5]See http://biotext.berkeley.edu/lql/ for a formal description of the language and additional examples.

Figure 1 illustrates the layered annotation of a sentence from biomedical text. Each annotation represents an interval spanning a sequence of characters, using absolute beginning and ending positions. Each layer corresponds to a conceptually different kind of annotation (e.g., word, gene/protein[6], shallow parse). Layers can be *sequential*, *overlapping* (e.g., two concepts sharing the same word), and *hierarchical* (either *spanning*, when the intervals are nested as in a parse tree, or *ontologically*, when the token itself is derived from a hierarchical ontology).

Word, POS and shallow parse layers are *sequential* (the latter can skip or span multiple words). The gene/protein layer assigns IDs from the LocusLink database of gene names.[7] For a given gene there are as many LocusLink IDs as the number of organisms it is found in (e.g., 4 in the case of the gene Bcl-2).

The MeSH layer contains entities from the hierarchical medical ontology MeSH (Medical Subject Headings).[8] The MeSH annotations on Figure 1 are overlapping (share the word *cell*) and hierarchical both ways: *spanning*, since *blood cell* (with MeSH id D001773) orthographically spans the word *cell* (id A11), and *ontologically*, since *blood cell* is a kind of *cell* and *cell death* (id D016923) is a kind of *Biological Phenomena*.

Given this annotation, we can extract potential protein-protein interactions from MEDLINE text. One simple approach is to follow (Blaschke et al., 1999), who developed a list of verbs (and their derived forms) and scanned for sentences containing the pattern PROTEIN ... INTERACTION-VERB ... PROTEIN. This can be expressed in LQL as follows:

```
FROM
[layer='sentence' { ALLOW GAPS }
  [layer='protein'] AS prot1
  [layer='pos' && tag_type="verb" &&
    content='activates']
  [layer='protein'] AS prot2
] SELECT prot1.content, prot2.content
```

This example extracts sentences containing a protein name in the gene/protein layer, followed by any sequence of words (because of ALLOW GAPS), followed by the interaction verb *activates*, followed by any sequence of words, and finally followed by an-

[6]Genes and their corresponding proteins often share the same name and the difference between them is often elided.

[7]http://www.ncbi.nlm.nih.gov/LocusLink

[8]http://www.nlm.nih.gov/mesh/meshhome.html

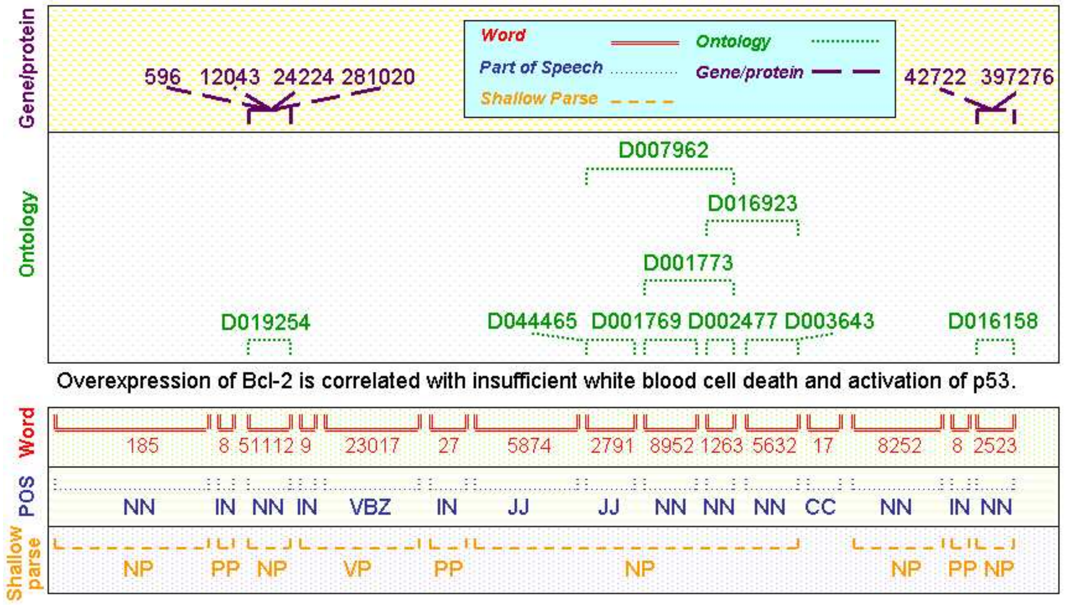


Figure 1: Illustration of the annotation layers. The *full parse*, *sentence* and *section* layers are not shown.

other protein name. All possible protein matches within the same sentence will be returned. The results are presented as pairs of protein names.

Each query level specifies a layer (e.g., sentence, part-of-speech, gene/protein) and optional restrictions on the attribute values. A binding statement is allowed after the layer's closing bracket. We can search for more than one verb simultaneously, e.g., by changing the POS layer of the query above to `[layer='pos' && (content='activates' || content='inhibit' || content='binds')]`. Further, a wildcard like `content ~ 'activate%'` can match the verb forms *activate*, *activates* and *activated*. We can also use double quotes `"` to make the comparison case insensitive. Finally, since LQL is automatically translated into SQL, SQL code can be written to surround the LQL query and to reference its results, thus allowing the use of SQL operators such as `GROUP BY`, `COUNT`, `DISTINCT`, `ORDER BY`, etc., as well as set operations like `UNION`.

Now consider the task of extracting interactions between chemicals and diseases. Given the sentence *"Adherence to statin prevents one coronary heart disease event for every 429 patients."*, we want to extract the relation that *statin* (potentially) prevents *coronary heart disease*. The latter is in the MeSH hierarchy (id D003327) with tree codes C14.280.647.250 and C14.907.553.470.250, while the former is listed in the MeSH supplementary concepts (ID C047068). In fact, the whole C subtree in MeSH contains diseases and all supplementary MeSH concepts represent chemicals. So we can find potentially useful sentences (to be further processed by another algorithm) using the following query:

```
FROM
[layer='sentence' {NO ORDER, ALLOW GAPS}
 [layer='shallow_parse' && tag_type='NP'
  [layer='chemicals'] AS chem $
 ]
 [layer='shallow_parse' && tag_type='NP'
  [layer='MeSH' && label BELOW "C"] AS dis $
 ]
] AS sent
SELECT chem.content,dis.content,sent.content
```

This looks for sentences containing two NPs in any order without overlaps (`NO ORDER`) and separated by any number of intervening elements. We further require one of the NPs to *end* (ensured by the `$` symbol) with a chemical, and the other (the disease) to end with a MeSH term from the C subtree.

## 4 System Architecture

Our basic model is similar to that of TIPSTER (Grishman, 1996): each annotation is stored as a record, which specifies the character-level beginning and ending positions, the layer and the type. The basic table[9] contains the following columns: **(1) annotation_id**; **(2) doc_id**; **(3) section**: *title*, *abstract* or *body*; **(4) layer_id**: layer identifier (*word*, *POS*, *shallow parse*, *sentence*, etc.); **(5) start_char_pos**: beginning character position, relative to *section* and *doc_id*; **(6) end_char_pos**: ending character position; **(7) tag_type**: a layer-specific token identifier.

After evaluating various different extensions of the structure above, we have arrived at one with some additional columns, which improves cross-layer query performance: **(8) sentence_id**; **(9) word_id**; **(10) first_word_pos**; and **(11) last_word_pos**. Columns (9)-(11) treat the *word* layer as atomic and require all annotations to coincide with word boundaries.

Finally, we use two types of *composite* indexes: *forward*, which looks for positions in a given document, and *inverted*, which supports searching based on annotation *values*.[10] An index lookup can be performed on any column combination that corresponds to an index *prefix*. An RDBMS' query optimizer estimates the optimal access paths (index and table scans), and join orders based on statistics collected over the stored records. In complex queries a combination of *forward* (F) and *inverted* (I) indexes is typically used. The particular ones we used are:[11]

(F) +doc_id+section+layer_id+sentence
+first_word_pos+last_word_pos+tag_type

(I) +layer_id+tag_type+doc_id+section+sentence
+first_word_pos+last_word_pos

(I) +word_id+layer_id+tag_type+doc_id+section
+sentence+first_word_pos

We have experimented with the system on a collection of 1.4 million MEDLINE abstracts, which include 10 million sentences annotated with 320 million multi-layered annotations. The current database size is around 70 GB. Annotations are indexed as they are inserted into the database.

[9]There are some additional tables mapping token IDs to entities (the string in case of a word, the MeSH label(s) in case of a MeSH term etc.)

[10]These *inverted* indexes can be seen as a direct extension of the widely used *inverted file* indexes in traditional IR systems.

[11]There is also an index on *annotation_id*, which allows for annotating relations between annotations.

Our initial evaluation shows variation in the execution time, depending on the kind and complexity of the query. Response time for simple queries is usually less than a minute, while for more complex ones it can be much longer. We are in the process of further investigating and tuning the system.

## 5 Conclusions and Future Work

We have provided a mechanism to effectively store and query layers of textual annotations, focusing on compact representation, easy query formulation, easy addition and removal of layers, and straightforward translation into SQL. Using a collection of 1.4 MEDLINE abstracts, we have evaluated various structures for data storage and have arrived at a promising one.

We have also designed a concise language (LQL) to express queries that span multiple levels of annotation structure, allowing users to express queries in a syntax that closely resembles the underlying annotation structure. We plan to release the software to the research community for use in their own annotation and querying needs.

**Acknowledgements** This research was supported by NSF DBI-0317510 and a gift from Genentech.

## References

Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(1-2):23–60.

Steven Bird, Yi Chen, Susan Davidson, Haejoong Lee, and Yifeng Zheng. 2005. Extending XPath to support linguistic queries. In *Proceedings of PLANX*, pages 35–46.

Christian Blaschke, Miguel Andrade, Christos Ouzounis, and Alfonso Valencia. 1999. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of ISMB*, pages 60–67.

Steve Cassidy and Jonathan Harrington. 2001. Multi-level annotation in the Emu speech database management system. *Speech Communication*, 33(1-2):61–77.

Ralph Grishman. 1996. Building an architecture: a CAWG saga. *Advances in Text Processing: Tipster Program Ph. II.*

Catherine Lai and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings Australasian Language Technology Workshop*, pages 139–146.

David McKelvie, Amy Isard, Andreas Mengel, Morten Moeller, Michael Grosse, and Marion Klein. 2001. The MATE workbench - an annotation tool for XML coded speech corpora. *Speech Communication*, 33(1-2):97–112.

# Word Alignment and Cross-Lingual Resource Acquisition [*]

**Carol Nichols and Rebecca Hwa**
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
{cln23,hwa}@cs.pitt.edu

## Abstract

Annotated corpora are valuable resources for developing Natural Language Processing applications. This work focuses on acquiring annotated data for multilingual processing applications. We present an annotation environment that supports a web-based user-interface for acquiring word alignments between English and Chinese as well as a visualization tool for researchers to explore the annotated data.

## 1 Introduction

The performance of many Natural Language Processing (NLP) applications can be improved through supervised machine learning techniques that train systems with annotated training examples. For example, a part-of-speech (POS) tagger might be induced from words that have been annotated with the correct POS tags. A limitation to the supervised approach is that the annotation is typically performed manually. This poses as a challenge in three ways. First, researchers must develop a comprehensive annotation guideline for the annotators to follow. Guideline development is difficult because researchers must be specific enough so that different annotators' work will be comparable, but also general enough to allow the annotators to make their own linguistic judgments. Reported experiences of previous annotation projects suggest that guideline development is both an art and a science and is itself a time-consuming process (Litman and Pan, 2002; Marcus et al., 1993; Xia et al., 2000; Wiebe, 2002). Second, it is common for the annotators to make mistakes, so some form of consistency check is necessary. Third, the entire process (guideline development, annotation, and error corrections) may have to be repeated with new domains.

This work focuses on the first two challenges: helping researchers to design better guidelines and to collect a large set of consistently labeled data from human annotators. Our annotation environment consists of two pieces of software: a user interface for the annotators and a visualization tool for the researchers to examine the data. The data-collection interface asks the users to make lexical and phrasal mappings (word alignments) between the two languages. Some studies suggest that supervised word aligned data may improve machine translation performance (Callison-Burch et al., 2004). The interface can also be configured to ask the annotators to correct projected annotated resources. The idea of projecting English annotation resources across word alignments has been explored in several studies (Yarowsky and Ngai, 2001; Hwa et al., 2005; Smith and Smith, 2004). Currently, our annotation interface is configured for correcting projected POS tagging for Chinese. The visualization tool aggregates the annotators' work, takes various statistics, and visually displays the aggregate information. Our goal is to aid the researchers conducting the experiment to identify noise in the annotations as well as problematic constructs for which the guidelines should provide further clarifications.

Our longer-term plan is to use this framework to support active learning (Cohn et al., 1996), a machine learning approach that aims to reduce the number of training examples needed by the system when it is provided with more informative training exam-

[*]This work has been supported, in part, by CRA-W Distributed Mentor Program. We thank Karina Ivanetich, David Chiang, and the NLP group at Pitt for helpful feedbacks on the user interfaces; Wanwan Zhang and Ying-Ju Suen for testing the system; and the anonymous reviewers for their comments on the paper.

ples. We believe that through a combination of an intuitive annotation interface, a visualization tool that checks for style and quality consistency, and appropriate active learning techniques, we can make supervised training more effective for developing multilingual applications.

## 2 Annotation Interface

One way to acquire annotations quickly is to appeal to users across the Internet. First, we are more likely to find annotators with the necessary qualifications. Second, many more users can work simultaneously than would be feasible to physically host in a lab. Third, having many users annotate the same data allows us to easily identify systematic problems as well as spurious mistakes. The OpenMind Initiative (Stork, 2001) has had success collecting information that could not be obtained from data mining tools or with a local small group of annotators.

Collecting data from users over the Internet introduces complications. Since we cannot ascertain the computer skills of the annotators, the interface must be easy to use. Our interface is a JAVA applet on a webpage so that it is platform independent. An online tutorial is also provided (and required for first-time users). Another problem of soliciting unknown users for data is the possibility of receiving garbage data created by users who do not have sufficient knowledge or are maliciously entering random input. Our system minimizes this risk in several ways. First, new users are required to work through the tutorial, which also serves as a short guide to reduce stylistic differences between the annotators. Second, we require the same data to be labeled by multiple people to ensure reliability, and researchers can use the visualization tool (see Section 3) to compare the agreement rates between annotators. Finally, our program is designed with a filter for malicious users. After completing the tutorial, the user is given a randomly selected sample sentence (for which we already have verified alignments) to annotate. The user must obtain an F-measure agreement of 60% with the "correct" alignments in order to be allowed to annotate sentences.[1]

Because word alignment annotation is a useful resource for both training and testing, quite a few interfaces have already been developed. The earliest is the Blinker Project (Melamed, 1998); more recent systems have been released to support more languages and visualization features (Ahrenberg et al., 2003; Lambert and Castell, 2004). [2] Our interface does share some similarities with these systems, but it is designed with additional features to support our experimental goals of guideline development, active learning and resource projection. Following the experimental design proposed by Och and Ney (2000), we instruct the annotators to indicate their level of confidence by choosing *sure* or *unsure* for each alignment they made. This allows researchers to identify areas where the translation may be unclear or difficult. We provide a text area for comments on each sentence so that the annotator may explain any assumptions or problems. A hidden timer records how long each user spends on each sentence in order to gauge the difficulty of the sentence; this information will be a useful measurement of the effectiveness of different active learning algorithms. Finally, our interface supports cross projection annotation. As an initial study, we have focused on POS tagging, but the framework can be extended for other types of resources such as syntactic and semantic trees and can be configured for languages other than English and Chinese. When words are aligned, the known and displayed English POS tag of the last English word involved in the alignment group is automatically projected onto all Chinese words involved, but a drop-down menu allows the user to correct this if the projection is erroneous. A screenshot of the interface is provided in Figure 1a.

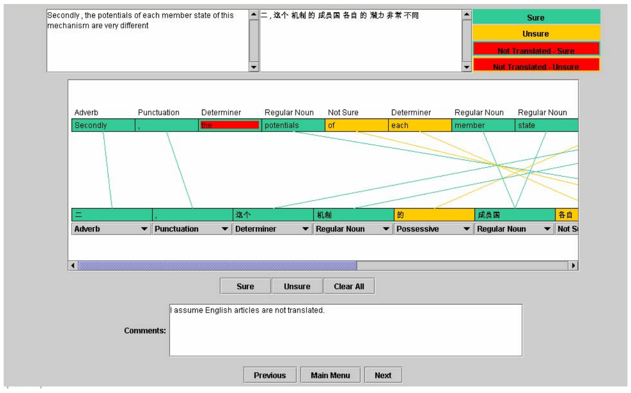
## 3 Tools for Researchers

Good training examples for NLP learning systems should have a high level of consistency and accuracy. We have developed a set of tools for researchers to visualize, compare, and analyze the work of the annotators. The main interface is a JAVA applet that provides a visual representation of all the alignments superimposed onto each other in a grid.

For the purposes of error detection, our system provides statistics for researchers to determine the agreement rates between the annotators. The metric we use is Cohen's K (1960), which is computed for every sentence across all users' alignments. Cohen's K is a measure of agreement that takes the total probability of agreement, subtracts the probability the agreement is due to chance, and divides by the maximum agreement possible. We use a variant of the

[1]The correct alignments were performed by two trained annotators who had an average agreement rate of about 85%. We chose 60% to be the figure of merit because this level is nearly impossible to obtain through random guessing but is lenient enough to allow for the inexperience of first time users. Automatic computer alignments average around 50%.

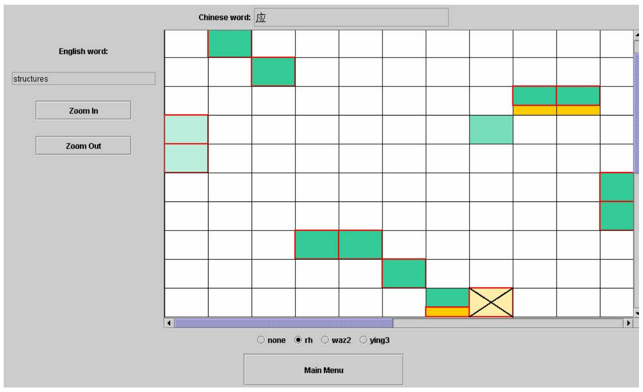
[2]Rada Mihalcea maintains an alignment resource repository (`http://www.cs.unt.edu/~rada/wa`) that contains other downloadable interface packages that do not have companion papers.


Secondly , the potentials of each member state of this mechanism are very different

二 , 这个 机制 的 成员国 各自 的 潜力 非常 不同

Comments: I assume English articles are not translated.

(a)


Chinese word: 应

English word: structures

( ) none [✓ rh] ( ) waz2 ( ) ying3

(b)

Figure 1: (a) A screenshot of the word alignment user-interface. (b) A screenshot of the visualization tool for analyzing multiple annotators' alignments.

equation that allows for having three or more judges (Davies and Fleiss, 1982). The measurement ranges from 0 (chance agreement) to 1 (perfect agreement). For any selected sentence, we also compute for each annotator an average pair-wise Cohen's K against all other users who aligned this sentence.[3] This statistic may be useful in several ways. First, someone with a consistently low score may not have enough knowledge to perform the task (or is malicious). Second, if an annotator received an unusually low score for a particular sentence, it might indicate that the person made mistakes in that sentence. Third, if there is too much disagreement among all users, the sentence might be a poor example to be included.

In addition to catching individual annotation errors, it is also important to minimize stylistic inconsistencies. These are differences in the ways different annotators (consistently) handle the same phenomena. A common scenario is that some function words in one language do not have an equivalent counterpart in the other language. Without a precise guideline ruling, some annotators always leave the function words unaligned while others always group the function words together with nearby content words. Our tool can be useful in developing and improving style guides. It highlights the potential areas that need further clarifications in the guidelines with an at-a-glance visual summary of where and how the annotators differed in their work. Each cell in the grid represents an alignment between one particular word in the English sentence and one particular word in the Chinese sentence. A white cell means no one proposed an alignment between the words. Each colored cell has two components: an upper green portion indicating a *sure* alignment and a lower yellow portion indicating an *unsure* alignment. The proportion of these components indicates the ratio of the number of people who marked this alignment as *sure* to those who were *unsure* (thus, an all-green cell means that everyone who aligned these words together is sure). Moreover, we use different saturation in the cells to indicate the percentage of people who aligned the two words together. A cell with faint colors means that most people did not chose to align these words together. Furthermore, researchers can elect to view the annotation decisions of a particular user by clicking on the radio buttons below. Only the selected user's annotation decisions would be highlighted by red outlines (i.e., only around the green portions of those cells that the person chose *sure* and around the yellow portions of this person's *unsure* alignments).

Figure 1b displays the result of three annotators' alignments of a sample sentence pair. This sentence seems reasonably easy to annotate. Most of the colored cells have a high saturation, showing that the annotators agree on the words to be aligned. Most of the cells are only green, showing that the annotators are sure of their decisions. Three out of the four *unsure* alignments coincide with the other annotators' *sure* alignments, and even in those cases, more annotators are sure than unsure (the green areas are 2/3 of the cells while the yellow areas are 1/3). The colored cells with low saturation indicate potential outliers. Comparing individual annotator's alignments against the composite, we find that one annotator, *rh*, may be a potential outlier annotator since this person generated the most number of lightly saturated cells. The person does not appear to be malicious since the three people's overall agreements are high. To determine whether the conflict

[3]not shown in the screenshot here.

arises from stylistic differences or from careless mistakes, researchers can click on the disputed cell (a cross will appear) to see the corresponding English and Chinese words in the text boxes in the top and left margin.

Different patterns in the visualization will indicate different problems. If the visualization patterns reveal a great deal of disagreement and *unsure* alignments overall, we might conclude that the sentence pair is a bad translation; if the disagreement is localized, this may indicate the presence of an idiom or a structure that does not translate word-for-word. Repeated occurrences of a pattern may suggest a stylistic inconsistency that should be addressed in the guidelines. Ultimately, each area of wide disagreement will require further analysis in order to determine which of these problems is occurring.

## 4 Conclusion and Future Work

In summary, we have presented an annotation environment for acquiring word alignments between English and Chinese as well as Part-Of-Speech tags for Chinese. The system is in place and the annotation process is underway.[4]

Once we have collected a medium-sized corpus, we will begin exploring different active learning techniques. Our goal is to find the best way to assign utility scores to the as-of-yet unlabeled sentences in order to obtain the greatest improvement in word alignment accuracy. Potential information useful for this task includes various measurements of the complexity of the sentence such as the rate of (automatic) alignments that are not one-to-one, the number of low-frequency words, and the number of potential language divergences (for example, many English verbs are nominalized in Chinese), and the co-occurrence of word pairs deemed to be *unsure* by the annotators in other contexts. Furthermore, we believe that the aggregate visualization tool will also help us uncover additional characteristics of potentially informative training examples.

[4]The annotation interface is open to public. Please visit `http://flan.cs.pitt.edu/~hwa/align/align.html`

## References

Lars Ahrenberg, Magnus Merkel, and Michael Petterstedt. 2003. Interactive word alignment for language engineering. In *Proceedings from EACL 2003*, Budapest.

Christopher Callison-Burch, David Talbot, and Miles Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, July.

J. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Meas.*, 20:37–46.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

M. Davies and J. Fleiss. 1982. Measuring agreement for multinomial data. *Biometrics*, 38:1047–1051.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Journal of Natural Language Engineering*. To appear.

Patrik Lambert and Nuria Castell. 2004. Alignment of parallel corpora exploiting asymmetrically aligned phrases. In *Proc. of the LREC 2004 Workshop on the Amazing Utility of Parallel and Comparable Corpora*, May.

Diane Litman and S. Pan. 2002. Desiging and evaluating an adaptive spoken dialogue system. *User Modeling and User-adapted Interaction*, 12(2/3):111–137.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

I. Dan Melamed. 1998. Annotation style guide for the blinker project. Technical Report IRCS 98-06, University of Pennsylvania.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using english to parse korean. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

David G Stork. 2001. Toward a computational theory of data acquisition and truthing. In *Proceedings of Computational Learning Theory (COLT 01).*

J. Wiebe. 2002. Instructions for annotating opinions in newspaper articles. Technical Report TR-02-101, University of Pittsburgh, Pittsburgh, PA.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Ocurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitch Marcus. 2000. Developing guidelines and ensuring consistency for chinese text annotation. In *Proceedings of the Second Language Resources and Evaluation Conference*, Athens, Greece, June.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the Second Meeting of the North American Association for Computational Linguistics*, pages 200–207.

# SenseRelate::TargetWord – A Generalized Framework for Word Sense Disambiguation

**Siddharth Patwardhan**
School of Computing
University of Utah
Salt Lake City, UT 84112
sidd@cs.utah.edu

**Satanjeev Banerjee**
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213
satanjeev@cmu.edu

**Ted Pedersen**
Dept. of Computer Science
University of Minnesota
Duluth, MN 55812
tpederse@d.umn.edu

## Abstract

We have previously introduced a method of word sense disambiguation that computes the intended sense of a target word, using WordNet-based measures of semantic relatedness (Patwardhan et al., 2003). *SenseRelate::TargetWord* is a Perl package that implements this algorithm. The disambiguation process is carried out by selecting that sense of the target word which is most related to the context words. Relatedness between word senses is measured using the *WordNet::Similarity* Perl modules.

## 1 Introduction

Many words have different meanings when used in different contexts. *Word Sense Disambiguation* is the task of identifying the intended meaning of a given *target word* from the context in which it is used. (Lesk, 1986) performed disambiguation by counting the number of overlaps between the dictionary definitions (i.e., glosses) of the target word and those of the neighboring words in the context. (Banerjee and Pedersen, 2002) extended this method of disambiguation by expanding the glosses of words to include glosses of related words, according to the structure of WordNet (Fellbaum, 1998). In subsequent work, (Patwardhan et al., 2003) and (Banerjee and Pedersen, 2003) proposed that measuring gloss overalps is just one way of determining *semantic relatedness*, and that word sense disambiguation can be performed by finding the most related sense of a target word to its surrounding context using a wide variety of measures of relatedness.

*SenseRelate::TargetWord* is a Perl package that implements these ideas, and is able to disambiguate a target word in context by finding the sense that is most related to its neighbors according to a specified measure. A user of this package is able to make a variety of choices for *text preprocessing options*, *context selection*, *relatedness measure selection* and the selection of an *algorithm* for computing the overall relatedness between each sense of the target word and words in the surrounding context. The user can customize each of these choices to fit the needs of her specific disambiguation task. Further, the various sub-tasks in the package are implemented in a modular fashion, allowing the user to easily replace a module with her own module if needed.

The following sections describe the generalized framework for Word Sense Disambiguation, the architecture and usage of *SenseRelate::TargetWord*, and a description of the user interfaces (command line and GUI).

## 2 The Framework

The package has a highly modular architecture. The disambiguation process is divided into a number of smaller sub-tasks, each of which is represented by a separate module. Each of the sequential sub-tasks or *stages* accepts data from a previous stage, performs a transformation on the data, and then passes on the processed data structures to the next stage in the pipeline. We have created a protocol that defines the structure and format of the data that is passed between the stages. The user can create her own

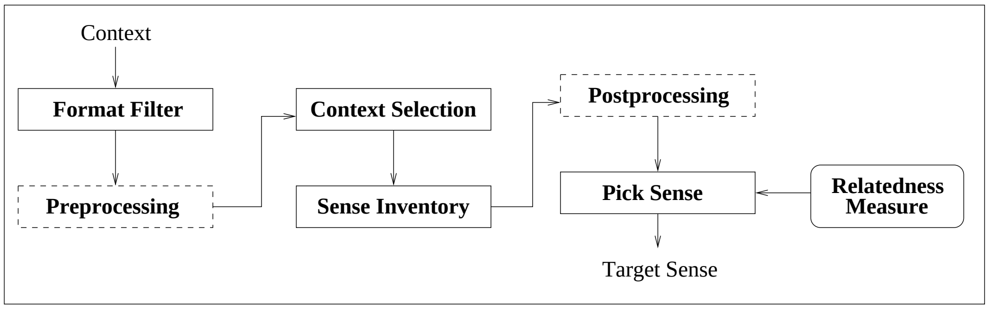


Figure 1: A generalized framework for *Word Sense Disambiguation*.

modules to perform any of these sub-tasks as long as the modules adhere to the protocol laid down by the package.

Figure 1 projects an overview of the architecture of the system and shows the various sub-tasks that need to be performed to carry out word sense disambiguation. The sub-tasks in the dotted boxes are optional. Further, each of the sub-tasks can be performed in a number of different ways, implying that the package can be customized in a large number of ways to suit different disambiguation needs.

### 2.1 Format Filter

The filter takes as input file(s) annotated in the SENSEVAL-2 lexical sample format, which is an XML–based format that has been used for both the SENSEVAL-2 and SENSEVAL-3 exercises. A file in this format includes a number of *instances*, each one made up of 2 to 3 lines of text where a single target word is designated with an XML tag. The filter parses the input file to build data structures that represent the instances to be disambiguated, which includes a single target word and the surrounding words that define the context.

### 2.2 Preprocessing

*SenseRelate::TargetWord* expects zero or more text preprocessing modules, each of which perform a transformation on the input words. For example, the *Compound Detection Module* identifies sequences of tokens that form compound words that are known as concepts to WordNet (such as “New York City”). In order to ensure that compounds are treated as a single unit, the package replaces them in the instance with the corresponding underscore–connected form (“New_York_City”).

Multiple preprocessing modules can be chained together, the output of one connected to the input of the next, to form a single preprocessing stage. For example, a part of speech tagging module could be added after compound detection.

### 2.3 Context Selection

Disambiguation is performed by finding the sense of the target word that is most related to the words in its surrounding context. The package allows for various methods of determining what exactly the surrounding context should consist of. In the current implementation, the context selection module uses an $n$ word window around the target word as context. The window includes the target word, and extends to both the left and right. The module selects the $n - 1$ words that are located closest to the target word, and sends these words (and the target) on to the next module for disambiguation. Note that these words must all be known to WordNet, and should not include any stop–words.

However, not all words in the surrounding context are indicative of the correct sense of the target word. An intelligent selection of the context words used in the disambiguation process could potentially yield much better results and generate a solution faster than if all the nearby words were used. For example, we could instead select the nouns from the window of context that have a high term–frequency to document–frequency ratio. Or, we could identify lexical chains in the surrounding context, and only include those words that are found in chains that include the target word.

### 2.4 Sense Inventory

After having reduced the context to $n$ words, the *Sense Inventory* stage determines the possible senses of each of the $n$ words. This list can be obtained from a dictionary, such as WordNet. A thesaurus could also be used for the purpose. Note however, that the subsequent modules in the pipeline should be aware of the codes assigned to the word senses.

In our system, this module first decides the base (uninflected) form of each of the $n$ words. It then retrieves all the senses for each word from the sense inventory. We use WordNet for our sense inventory.

### 2.5 Postprocessing

Some optional processing can be performed on the data structures generated by the Sense Inventory module. This would include tasks such as *sense pruning*, which is the process of removing some senses from the inventory, based on simple heuristics, algorithms or options. For example, the user may decide to preclude all verb senses of the target word from further consideration in the disambiguation process.

### 2.6 Identifying the Sense

The disambiguation module takes the lists of senses of the target word and those of the context words and uses this information to pick one sense of the target word as the answer. Many different algorithms could be used to do this. We have modules *Local* and *Global* that (in different ways) determine the relatedness of each of the senses of the target word with those of the context words, and pick the most related sense as the answer. These are described in greater detail by (Banerjee and Pedersen, 2002), but in general the Local method compares the target word to its neighbors in a pair-wise fashion, while the Global method carries out an exhaustive comparison between all the senses of the target word and all the senses of the neighbors.

## 3 Using SenseRelate::TargetWord

*SenseRelate::TargetWord* can be used via the command-line interface provided by the utility program called *disamb.pl*. It provides a rich variety of options for controlling the process of disambiguation. Or, it can be embedded into Perl programs, by including it as a module and calling its various methods. Finally, there is a graphical interface to the package that allows a user to highlight a word in context to be disambiguated.

### 3.1 Command Line

The command-line interface *disamb.pl* takes as input a SENSEVAL-2 formatted lexical sample file. The program disambiguates the marked up word in each instance and prints to screen the instance ID, along with the disambiguated sense of the target word.

Command line options are available to control the disambiguation process. For example, a user can specify which relatedness measure they would like to use, whether disambiguation should be carried out using Local or Global methods, how large a window of context around the target word is to be used, and whether or not all the parts of speech of a word should be considered.

### 3.2 Programming Interface

*SenseRelate::TargetWord* is distributed as a Perl package. It is programmed in object-oriented Perl as a group of Perl classes. Objects of these classes can be instantiated in user programs, and methods can be called on these objects. The package requires that the Perl interface to WordNet, *WordNet::QueryData*[1] be installed on the system. The disambiguation algorithms also require that the semantic relatedness measures *WordNet::Similarity* (Pedersen et al., 2004) be installed.

### 3.3 Graphical User Interface

We have developed a graphical interface for the package in order to conveniently access the disambiguation modules. The GUI is written in Gtk-Perl – a Perl API to the Gtk toolkit. Unlike the command line interface, the graphical interface is not tied to any input file format. The interface allows the user to input text, and to select the word to disambiguate. It also provides the user with numerous configuration options corresponding to the various customizations described above.

[1]http://search.cpan.org/dist/WordNet-QueryData

## 4 Related Work

There is a long history of work in Word Sense Disambiguation that uses Machine Readable Dictionaries, and are highly related to our approach.

One of the first approaches was that of (Lesk, 1986), which treated every dictionary definition of a concept as a bag of words. To identify the intended sense of the target word, the Lesk algorithm would determine the number of word overlaps between the definitions of each of the meanings of the target word, and those of the context words. The meaning of the target word with maximum definition overlap with the context words was selected as the intended sense.

(Wilks et al., 1993) developed a context vector approach for performing word sense disambiguation. Their algorithm built co-occurrence vectors from dictionary definitions using Longman's Dictionary of Contemporary English (LDOCE). They then determined the extent of overlap between the sum of the vectors of the words in the context and the sum of the vectors of the words in each of the definitions (of the target word). For vectors, the extent of overlap is defined as the dot product of the vectors. The meaning of the target word that had the maximum overlap was selected as the answer.

More recently, (McCarthy et al., 2004) present a method that performs disambiguation by determing the most frequent sense of a word in a particular domain. This is based on measuring the relatedness of the different possible senses of a target word (using *WordNet::Similarity*) to a set of words associated with a particular domain that have been identified using distributional methods. The relatedness scores between a target word and the members of this set are scaled by the distributional similarity score.

## 5 Availability

*SenseRelate::TargetWord* is written in Perl and is freely distributed under the Gnu Public License. It is available via SourceForge, an Open Source development platform[2], and the Comprehensive Perl Archive Network (CPAN)[3].

[2]http://senserelate.sourceforge.net

[3]http://search.cpan.org/dist/WordNet-SenseRelate-TargetWord

## 6 Acknowledgements

This research is partially supported by a National Science Foundation Faculty Early CAREER Development Award (#0092784).

## References

S. Banerjee and T. Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City, February.

S. Banerjee and T. Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico, August.

C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

M. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from a ice cream cone. In *Proceedings of SIGDOC '86*.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 279–286, Barcelona, Spain, July.

S. Patwardhan, S. Banerjee, and T. Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING-03)*, Mexico City, Mexico, February.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::Similarity - Measuring the Relatedness of Concepts. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Y. Wilks, D. Fass, C. Guo, J. McDonald, T. Plate, and B. Slator. 1993. Providing machine tractable dictionary tools. In J. Pustejovsky, editor, *Semantics and the Lexicon*. Kluwer Academic Press, Dordrecht and Boston.

# A Practical Solution to the Problem of Automatic Part-of-Speech Induction from Text

**Reinhard Rapp**
University of Mainz, FASK
D-76711 Germersheim, Germany
rapp@mail.fask.uni-mainz.de

## Abstract

The problem of part-of-speech induction from text involves two aspects: Firstly, a set of word classes is to be derived automatically. Secondly, each word of a vocabulary is to be assigned to one or several of these word classes. In this paper we present a method that solves both problems with good accuracy. Our approach adopts a mixture of statistical methods that have been successfully applied in word sense induction. Its main advantage over previous attempts is that it reduces the syntactic space to only the most important dimensions, thereby almost eliminating the otherwise omnipresent problem of data sparseness.

## 1 Introduction

Whereas most previous statistical work concerning parts of speech has been on tagging, this paper deals with part-of-speech induction. In part-of-speech induction two phases can be distinguished: In the first phase a set of word classes is to be derived automatically on the basis of the distribution of the words in a text corpus. These classes should be in accordance with human intuitions, i.e. common distinctions such as nouns, verbs and adjectives are desirable. In the second phase, based on its observed usage each word is assigned to one or several of the previously defined classes.

The main reason why part-of-speech induction has received far less attention than part-of-speech tagging is probably that there seemed no urgent need for it as linguists have always considered classifying words as one of their core tasks, and as a consequence accurate lexicons providing such information are readily available for many languages. Nevertheless, deriving word classes automatically is an interesting intellectual challenge with relevance to cognitive science. Also, advantages of the automatic systems are that they should be more objective and can provide precise information on the likelihood distribution for each of a word's parts of speech, an aspect that is useful for statistical machine translation.

The pioneering work on class based n-gram models by Brown et al. (1992) was motivated by such considerations. In contrast, Schütze (1993) by applying a neural network approach put the emphasis on the cognitive side. More recent work includes Clark (2003) who combines distributional and morphological information, and Freitag (2004) who uses a hidden Marcov model in combination with co-clustering.

Most studies use abstract statistical measures such as perplexity or the F-measure for evaluation. This is good for quantitative comparisons, but makes it difficult to check if the results agree with human intuitions. In this paper we use a straightforward approach for evaluation. It involves checking if the automatically generated word classes agree with the word classes known from grammar books, and whether the class assignments for each word are correct.

## 2 Approach

In principle, word classification can be based on a number of different linguistic principles, e.g. on phonology, morphology, syntax or semantics. However, in this paper we are only interested in syntactically motivated word classes. With syntactic classes the aim is that words belonging to the same class can substitute for one another in a sentence without affecting its grammaticality.

As a consequence of the substitutability, when looking at a corpus words of the same class typically have a high agreement concerning their left and right neighbors. For example, nouns are frequently preceded by words like *a*, *the*, or *this*, and succeeded by words like *is*, *has* or *in*. In statistical terms, words of the same class have a similar frequency distribution concerning their left and right neighbors. To some extend this can also be observed with indirect neighbors, but with them the effect is less salient and therefore we do not consider them here.

The co-occurrence information concerning the words in a vocabulary and their neighbors can be stored in a matrix as shown in table 1. If we now want to discover word classes, we simply compute the similarities between all pairs of rows using a vector similarity measure such as the cosine coefficient and then cluster the words according to these similarities. The expectation is that unambiguous nouns like *breath* and *meal* form one cluster, and that unambiguous verbs like *discuss* and *protect* form another cluster.

Ambiguous words like *link* or *suit* should not form a tight cluster but are placed somewhere in between the noun and the verb clusters, with the exact position depending on the ratios of the occurrence frequencies of their readings as either a noun or a verb. As this ratio can be arbitrary, according to our experience ambiguous words do not severely affect the clustering but only form some uniform background noise which more or less cancels out in a large vocabulary.[1] Note that the correct assignment of the ambiguous words to clusters is not required at this stage, as this is taken care of in the next step.

This step involves computing the differential vector of each word from the centroid of its closest cluster, and to assign the differential vector to the most appropriate other cluster. This process can be repeated until the length of the differential vector falls below a threshold or, alternatively, the agreement with any of the centroids becomes too low. This way an ambiguous word is assigned to several parts of speech, starting from the most common and proceeding to the least common. Figure 1 illustrates this process.

[1] An alternative to relying on this fortunate but somewhat unsatisfactory effect would be not to use global co-occurrence vectors but local ones, as successfully proposed in word sense induction (Rapp, 2004). This means that every occurrence of a word obtains a separate row vector in table 1. The problem with the resulting extremely sparse matrix is that most vectors are either orthogonal to each other or duplicates of some other vector, with the consequence that the dimensionality reduction that is indispensable for such matrices does not lead to sensible results. This problem is not as severe in word sense induction where larger context windows are considered.

The procedure that we described so far works in theory but not well in practice. The problem with it is that the matrix is so sparse that sampling errors have a strong negative effect on the results of the vector comparisons. Fortunately, the problem of data sparseness can be minimized by reducing the dimensionality of the matrix. An appropriate algebraic method that has the capability to reduce the dimensionality of a rectangular matrix is *Singular Value Decomposition* (SVD). It has the property that when reducing the number of columns the similarities between the rows are preserved in the best possible way. Whereas in other studies the reduction has typically been from several ten thousand to a few hundred, our reduction is from several ten thousand to only three. This leads to a very strong generalization effect that proves useful for our particular task.

| | left neighbors | | | | right neighbors | | | |
|---|---|---|---|---|---|---|---|---|
| | a | we | the | you | a | can | is | well |
| breath | 11 | 0 | 18 | 0 | 0 | 14 | 19 | 0 |
| discuss | 0 | 17 | 0 | 10 | 9 | 0 | 0 | 8 |
| link | 14 | 6 | 11 | 7 | 10 | 9 | 14 | 3 |
| meal | 15 | 0 | 17 | 0 | 0 | 9 | 12 | 0 |
| protect | 0 | 15 | 1 | 12 | 14 | 0 | 0 | 4 |
| suit | 5 | 0 | 8 | 3 | 0 | 8 | 16 | 2 |

Table 1. Co-occurrence matrix of adjacent words.

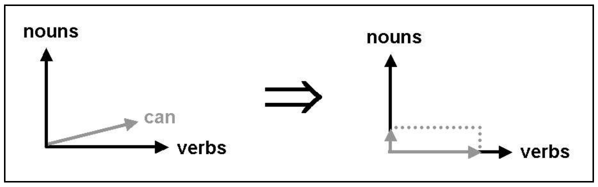


Figure 1. Constructing the parts of speech for *can*.

## 3 Procedure

Our computations are based on the unmodified text of the 100 million word *British National Corpus* (BNC), i.e. including all function words and without lemmatization. By counting the occurrence frequencies for pairs of adjacent words we compiled a matrix as exemplified in table 1. As this matrix is too large to be processed with our algorithms (SVD and clustering), we decided to restrict the number of rows to a vocabulary appropriate for evaluation purposes. Since we are not aware of any standard vocabulary previously used in related work, we manually selected an ad hoc list of 50 words with BNC frequencies between 5000 and 6000 as shown in table 2. The choice of 50 was motivated by the intention to give complete clustering results in graphical form. As we did not want to deal with morphology, we used base forms only. Also, in order to be able to subjectively judge the results, we only selected words where we felt reasonably confident about their possible parts of speech. Note that the list of words was compiled before the start of our experiments and remained unchanged thereafter.

The co-occurrence matrix based on the restricted vocabulary and all neighbors occurring in the BNC has a size of 50 rows times 28,443 columns. As our transformation function we simply use the logarithm after adding one to each value in the matrix.[2] As usual, the one is added for smoothing purposes and to avoid problems with zero values. We decided not to use a sophisticated association measure such as the log-likelihood ratio because it has an inappropriate value characteristic that prevents the SVD, which is conducted in the next step, from finding optimal dimensions.[3]

The purpose of the SVD is to reduce the number of columns in our matrix to the main dimensions. However, it is not clear how many dimensions should be computed. Since our aim of identifying basic word classes such as nouns or verbs requires strong generalizations instead of subtle distinctions, we decided to take only the three main dimensions into account, i.e. the resulting matrix has a size of 50 rows times 3 columns.[4] The last step in our procedure involves applying a clustering algorithm to the 50 words corresponding to the rows in the matrix. We used hierarchical clustering with average linkage, a linkage type that provides considerable tolerance concerning outliers.

## 4 Results and Evaluation

Our results are presented as dendrograms which in contrast to 2-dimensional dot-plots have the advantage of being able to correctly show the true distances between clusters. The two dendrograms in figure 2 where both computed by applying the procedure as described in the previous section, with the only difference that in generating the upper dendrogram the SVD-step has been omitted, whereas in generating the lower dendrogram it has been conducted. Without SVD the expected clusters of verbs, nouns and adjectives are not clearly separated, and the adjectives *widely* and *rural* are placed outside the adjective cluster. With SVD, all 50 words are in their appropriate clusters and the three discovered clusters are much more salient. Also, *widely* and *rural* are well within the adjective cluster. The comparison of the two dendrograms indicates that the SVD was capable of making appropriate generalizations. Also, when we look inside each cluster we can see that ambiguous words like *suit*, *drop* or *brief* are somewhat closer to their secondary class than unambiguous words.

Having obtained the three expected clusters, the next investigation concerns the assignment of the ambiguous words to additional clusters. As described previously, this is done by computing differential vectors, and by assigning these to the most similar other cluster. Hereby for the cosine similarity we set a threshold of 0.8. That is, only if the similarity between the differential vector and its closest centroid was higher than 0.8 we assigned the word to this cluster and continued to compute differential vectors. Otherwise we assumed that the differential vector was caused by sampling errors and aborted the process of searching for additional class assignments.

The results from this procedure are shown in table 2 where for each of the 50 words all computed classes are given in the order as they were obtained by the algorithm, i.e. the dominant assignments are listed first. Although our algorithm does not name the classes, for simplicity we interpret them in the obvious way, i.e. as nouns, verbs and adjectives. A comparison with WordNet 2.0 choices is given in brackets. For example, +N means that WordNet lists the additional assignment *noun*, and -A indicates that the assignment *adjective* found by the algorithm is not listed in WordNet.

According to this comparison, for all 50 words the first reading is correct. For 16 words an additional second reading was computed which is correct in 11 cases. 16 of the WordNet assignments are missing, among them the verb readings for *reform*, *suit*, and *rain* and the noun reading for *serve*. However, as many of the WordNet assignments seem rare, it is not clear in how far the omissions can be attributed to shortcomings of the algorithm.

[2] For arbitrary vocabularies the row vectors should be divided by the corpus frequency of the corresponding word.
[3] We are currently investigating if replacing the log-likelihood values by their ranks can solve this problem.
[4] Note that larger matrices can require a few more dimensions.

| | | |
|---|---|---|
| accident N | expensive A | reform N (+V) |
| belief N | familiar A (+N) | rural A |
| birth N (+V) | finance N V | screen N (+V) |
| breath N | grow V N (-N) | seek V (+N) |
| brief A N | imagine V | serve V (+N) |
| broad A (+N) | introduction N | slow A V |
| busy A V | link N V | spring N A V (-A) |
| catch V N | lovely A (+N) | strike N V |
| critical A | lunch N (+V) | suit N (+V) |
| cup N (+V) | maintain V | surprise N V |
| dangerous A | occur V N (-N) | tape N V |
| discuss V | option N | thank V A (-A) |
| drop V N | pleasure N | thin A (+V) |
| drug N (+V) | protect V | tiny A |
| empty A V (+N) | prove V | widely A N (-N) |
| encourage V | quick A (+N) | wild A (+N) |
| establish V | rain N (+V) | |

Table 2. Computed parts of speech for each word.

## 5 Summary and Conclusions

This work was inspired by previous work on word sense induction. The results indicate that part of speech induction is possible with good success based on the analysis of distributional patterns in text. The study also gives some insight how SVD is capable of significantly improving the results.

Whereas in a previous paper (Rapp, 2004) we found that for word sense induction the local clustering of local vectors is more appropriate than the global clustering of global vectors, for part-of-speech induction our conclusion is that the situation is exactly the other way round, i.e. the global clustering of global vectors is more adequate (see footnote 1). This finding is of interest when trying to understand the nature of syntax versus semantics if expressed in statistical terms.

## Acknowledgements

I would like to thank Manfred Wettler and Christian Biemann for comments, Hinrich Schütze for the SVD-software, and the DFG (German Research Society) for financial support.

## References

Brown, Peter F.; Della Pietra, Vincent J.; deSouza, Peter V.; Lai, Jennifer C.; Mercer, Robert L. (1992). Class-based n-gram models of natural language. *Computational Linguistics* 18(4), 467-479.

Clark, Alexander (2003). Combining distributional and morphological information for part of speech induction. *Proceedings of 10th EACL*, Budapest, *59-66.*

Freitag, Dayne (2004). Toward unsupervised whole-corpus tagging. *Proceedings of COLING*, Geneva, 357-363.

Rapp, Reinhard (2004). A practical solution to the problem of automatic word sense induction. *Proceedings of ACL (Companion Volume)*, Barcelona, 195-198.

Schütze, Hinrich (1993). Part-of-speech induction from scratch. *Proceedings of ACL*, Columbus, 251-258.

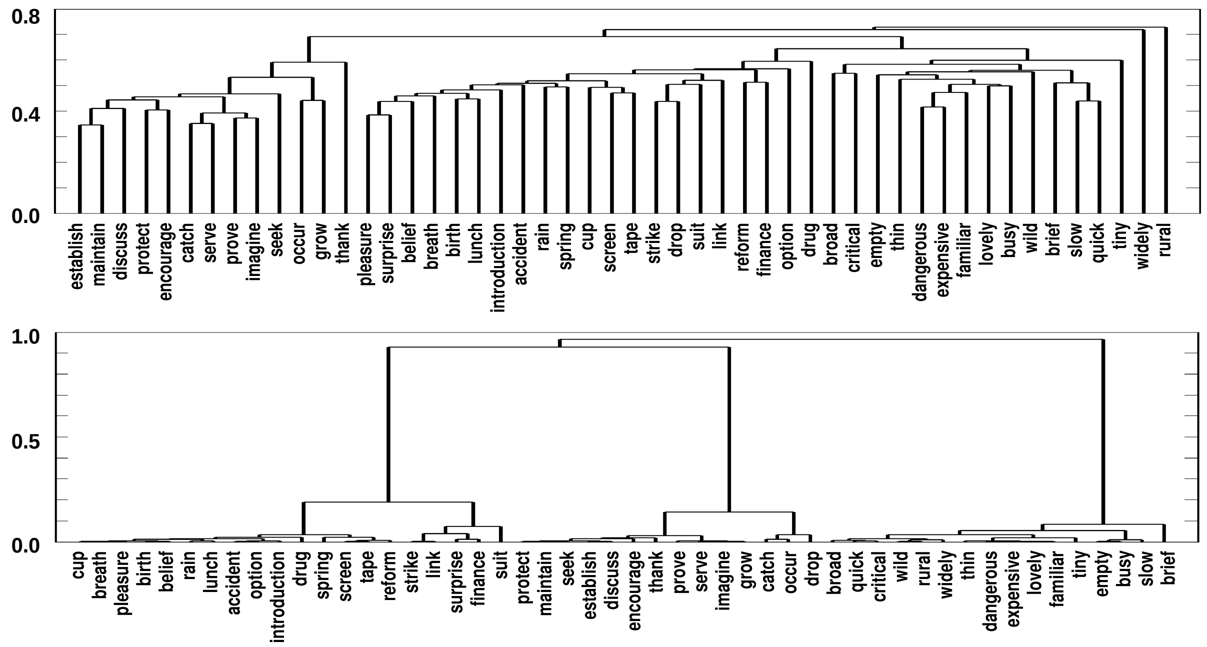


Figure 2. Syntactic similarities with (lower dendrogram) and without SVD (upper dendrogram).

# Automating Temporal Annotation with TARSQI

**Marc Verhagen**[†], **Inderjeet Mani**[‡], **Roser Sauri**[†],
**Robert Knippen**[†], **Seok Bae Jang**[‡], **Jessica Littman**[†],
**Anna Rumshisky**[†], **John Phillips**[‡], **James Pustejovsky**[†]

† Department of Computer Science, Brandeis University, Waltham, MA 02254, USA
{marc,roser,knippen,jlittman,arum,jamesp}@cs.brandeis.edu
‡ Computational Linguistics, Georgetown University, Washington DC, USA
{im5,sbj3,jbp24}@georgetown.edu

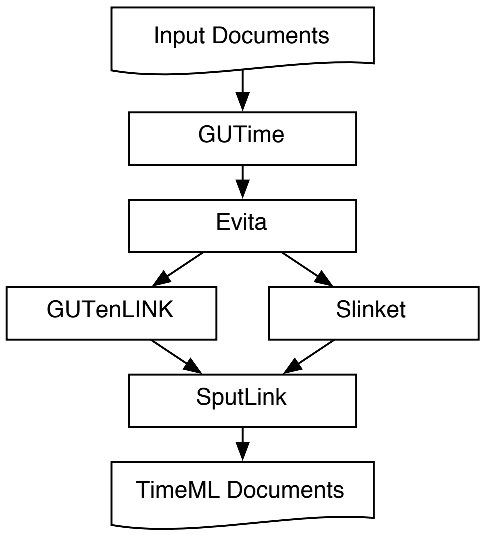
## Abstract

We present an overview of TARSQI, a modular system for automatic temporal annotation that adds time expressions, events and temporal relations to news texts.

## 1 Introduction

The TARSQI Project (Temporal Awareness and Reasoning Systems for Question Interpretation) aims to enhance natural language question answering systems so that temporally-based questions about the events and entities in news articles can be addressed appropriately. In order to answer those questions we need to know the temporal ordering of events in a text. Ideally, we would have a total ordering of all events in a text. That is, we want an event like *marched* in *ethnic Albanians marched Sunday in downtown Istanbul* to be not only temporally related to the nearby time expression *Sunday* but also ordered with respect to all other events in the text. We use TimeML (Pustejovsky et al., 2003; Saurí et al., 2004) as an annotation language for temporal markup. TimeML marks time expressions with the TIMEX3 tag, events with the EVENT tag, and temporal links with the TLINK tag. In addition, syntactic subordination of events, which often has temporal implications, can be annotated with the SLINK tag.

A complete manual TimeML annotation is not feasible due to the complexity of the task and the sheer amount of news text that awaits processing. The TARSQI system can be used stand-alone or as a means to alleviate the tasks of human annotators. Parts of it have been intergrated in Tango, a graphical annotation environment for event ordering (Verhagen and Knippen, Forthcoming). The system is set up as a cascade of modules that successively add more and more TimeML annotation to a document. The input is assumed to be part-of-speech tagged and chunked. The overall system architecture is laid out in the diagram below.



In the following sections we describe the five TARSQI modules that add TimeML markup to news texts.

## 2 GUTime

The GUTime tagger, developed at Georgetown University, extends the capabilities of the TempEx tagger (Mani and Wilson, 2000). TempEx, developed at MITRE, is aimed at the ACE TIMEX2 standard (timex2.mitre.org) for recognizing the extents and normalized values of time expressions. TempEx handles both absolute times (e.g., *June 2, 2003*) and relative times (e.g., *Thursday*) by means of a number of tests on the local context. Lexical triggers like *today*, *yesterday*, and *tomorrow*, when used in a specific sense, as well as words which indicate a positional offset, like *next month*, *last year*, *this coming Thursday* are resolved based on computing direction and magnitude with respect to a reference time, which is usually the document publication time.

GUTime extends TempEx to handle time expressions based on the TimeML TIMEX3 standard (timeml.org), which allows a functional style of encoding offsets in time expressions. For example, *last week* could be represented not only by the time value but also by an expression that could be evaluated to compute the value, namely, that it is the week preceding the week of the document date. GUTime also handles a variety of ACE TIMEX2 expressions not covered by TempEx, including durations, a variety of temporal modifiers, and European date formats. GUTime has been benchmarked on training data from the Time Expression Recognition and Normalization task (timex2.mitre.org/tern.html) at .85, .78, and .82 F-measure for timex2, text, and val fields respectively.

## 3 EVITA

Evita (Events in Text Analyzer) is an event recognition tool that performs two main tasks: robust event identification and analysis of grammatical features, such as tense and aspect. Event identification is based on the notion of event as defined in TimeML. Different strategies are used for identifying events within the categories of verb, noun, and adjective. Event identification of verbs is based on a lexical look-up, accompanied by a minimal contextual parsing, in order to exclude weak stative predicates such as *be* or *have*. Identifying events expressed by nouns, on the other hand, involves a disambiguation phase in addition to lexical lookup. Machine learning techniques are used to determine when an ambiguous noun is used with an event sense. Finally, identifying adjectival events takes the conservative approach of tagging as events only those adjectives that have been lexically pre-selected from TimeBank[1], whenever they appear as the head of a predicative complement. For each element identified as denoting an event, a set of linguistic rules is applied in order to obtain its temporally relevant grammatical features, like tense and aspect. Evita relies on preprocessed input with part-of-speech tags and chunks. Current performance of Evita against TimeBank is .75 precision, .87 recall, and .80 F-measure. The low precision is mostly due to Evita's over-generation of generic events, which were not annotated in TimeBank.

## 4 GUTenLINK

Georgetown's GUTenLINK TLINK tagger uses hand-developed syntactic and lexical rules. It handles three different cases at present: (i) the event is anchored without a signal to a time expression within the same clause, (ii) the event is anchored without a signal to the document date speech time frame (as in the case of reporting verbs in news, which are often at or offset slightly from the speech time), and (iii) the event in a main clause is anchored with a signal or tense/aspect cue to the event in the main clause of the previous sentence. In case (iii), a finite state transducer is used to infer the likely temporal relation between the events based on TimeML tense and aspect features of each event. For example, a past tense non-stative verb followed by a past perfect non-stative verb, with grammatical aspect maintained, suggests that the second event precedes the first.

GUTenLINK uses default rules for ordering events; its handling of successive past tense non-stative verbs in case (iii) will not correctly order sequences like *Max fell. John pushed him*. GUTenLINK is intended as one component in a larger machine-learning based framework for ordering events. Another component which will be developed will leverage document-level inference, as in the machine learning approach of (Mani et al., 2003), which required annotation of a reference time (Reichenbach, 1947; Kamp and Reyle, 1993) for the event in each finite clause.

[1]TimeBank is a 200-document news corpus manually annotated with TimeML tags. It contains about 8000 events, 2100 time expressions, 5700 TLINKs and 2600 SLINKs. See (Day et al., 2003) and www.timeml.org for more details.

An early version of GUTenLINK was scored at .75 precision on 10 documents. More formal Precision and Recall scoring is underway, but it compares favorably with an earlier approach developed at Georgetown. That approach converted event-event TLINKs from TimeBank 1.0 into feature vectors where the TLINK relation type was used as the class label (some classes were collapsed). A C5.0 decision rule learner trained on that data obtained an accuracy of .54 F-measure, with the low score being due mainly to data sparseness.

## 5 Slinket

Slinket (SLINK Events in Text) is an application currently being developed. Its purpose is to automatically introduce SLINKs, which in TimeML specify subordinating relations between pairs of events, and classify them into factive, counterfactive, evidential, negative evidential, and modal, based on the modal force of the subordinating event. Slinket requires chunked input with events.

SLINKs are introduced by a well-delimited subgroup of verbal and nominal predicates (such as *regret*, *say*, *promise* and *attempt*), and in most cases clearly signaled by the context of subordination. Slinket thus relies on a combination of lexical and syntactic knowledge. Lexical information is used to pre-select events that may introduce SLINKs. Predicate classes are taken from (Kiparsky and Kiparsky, 1970; Karttunen, 1971; Hooper, 1975) and subsequent elaborations of that work, as well as induced from the TimeBank corpus. A syntactic module is applied in order to properly identify the subordinated event, if any. This module is built as a cascade of shallow syntactic tasks such as clause boundary recognition and subject and object tagging. Such tasks are informed from both linguistic-based knowledge (Papageorgiou, 1997; Leffa, 1998) and corpora-induced rules (Sang and Déjéan, 2001); they are currently being implemented as sequences of finite-state transducers along the lines of (Aït-Mokhtar and Chanod, 1997). Evaluation results are not yet available.

## 6 SputLink

SputLink is a temporal closure component that takes known temporal relations in a text and derives new implied relations from them, in effect making explicit what was implicit. A temporal closure component helps to find those global links that are not necessarily derived by other means. SputLink is based on James Allen's interval algebra (1983) and was inspired by (Setzer, 2001) and (Katz and Arosio, 2001) who both added a closure component to an annotation environment.

Allen reduces all events and time expressions to intervals and identifies 13 basic relations between the intervals. The temporal information in a document is represented as a graph where events and time expressions form the nodes and temporal relations label the edges. The SputLink algorithm, like Allen's, is basically a constraint propagation algorithm that uses a transitivity table to model the compositional behavior of all pairs of relations. For example, if A precedes B and B precedes C, then we can compose the two relations and infer that A precedes C. Allen allowed unlimited disjunctions of temporal relations on the edges and he acknowledged that inconsistency detection is not tractable in his algebra. One of SputLink's aims is to ensure consistency, therefore it uses a restricted version of Allen's algebra proposed by (Vilain et al., 1990). Inconsistency detection is tractable in this restricted algebra.

A SputLink evaluation on TimeBank showed that SputLink more than quadrupled the amount of temporal links in TimeBank, from 4200 to 17500. Moreover, closure adds non-local links that were systematically missed by the human annotators. Experimentation also showed that temporal closure allows one to structure the annotation task in such a way that it becomes possible to create a complete annotation from local temporal links only. See (Verhagen, 2004) for more details.

## 7 Conclusion and Future Work

The TARSQI system generates temporal information in news texts. The five modules presented here are held together by the TimeML annotation language and add time expressions (GUTime), events (Evita), subordination relations between events (Slinket), local temporal relations between times and events (GUTenLINK), and global temporal relations between times and events (SputLink).

In the nearby future, we will experiment with more strategies to extract temporal relations from texts. One avenue is to exploit temporal regularities in SLINKs, in effect using the output of Slinket as a means to derive even more TLINKs. We are also compiling more annotated data in order to provide more training data for machine learning approaches to TLINK extraction. SputLink currently uses only qualitative temporal infomation, it will be extended to use quantitative information, allowing it to reason over durations.

## References

Salah Aït-Mokhtar and Jean-Pierre Chanod. 1997. Subject and Object Dependency Extraction Using Finite-State Transducers. In *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications. ACL/EACL-97 Workshop Proceedings*, pages 71–77, Madrid, Spain. Association for Computational Linguistics.

James Allen. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843.

David Day, Lisa Ferro, Robert Gaizauskas, Patrick Hanks, Marcia Lazo, James Pustejovsky, Roser Saurí, Andrew See, Andrea Setzer, and Beth Sundheim. 2003. The TimeBank Corpus. *Corpus Linguistics*.

Joan Hooper. 1975. On Assertive Predicates. In John Kimball, editor, *Syntax and Semantics*, volume IV, pages 91–124. Academic Press, New York.

Hans Kamp and Uwe Reyle, 1993. *From Discourse to Logic*, chapter 5, Tense and Aspect, pages 483–546. Kluwer Academic Publishers, Dordrecht, Netherlands.

Lauri Karttunen. 1971. Some Observations on Factivity. In *Papers in Linguistics*, volume 4, pages 55–69.

Graham Katz and Fabrizio Arosio. 2001. The Annotation of Temporal Information in Natural Language Sentences. In *Proceedings of ACL-EACL 2001, Workshop for Temporal and Spatial Information Processing*, pages 104–111, Toulouse, France. Association for Computational Linguistics.

Paul Kiparsky and Carol Kiparsky. 1970. Fact. In Manfred Bierwisch and Karl Erich Heidolph, editors, *Progress in Linguistics. A collection of Papers*, pages 143–173. Mouton, Paris.

Vilson Leffa. 1998. Clause Processing in Complex Sentences. In *Proceedings of the First International Conference on Language Resources and Evaluation*, volume 1, pages 937–943, Granada, Spain. ELRA.

Inderjeet Mani and George Wilson. 2000. Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL2000)*, pages 69–76.

Inderjeet Mani, Barry Schiffman, and Jianping Zhang. 2003. Inferring Temporal Ordering of Events in News. Short Paper. In *Proceedings of the Human Language Technology Conference (HLT-NAACL'03)*.

Harris Papageorgiou. 1997. Clause Recognition in the Framework of Allignment. In Ruslan Mitkov and Nicolas Nicolov, editors, *Recent Advances in Natural Language Recognition*. John Benjamins, Amsterdam, The Netherlands.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS-5 Fifth International Workshop on Computational Semantics*.

Hans Reichenbach. 1947. *Elements of Symbolic Logic*. MacMillan, London.

Tjong Kim Sang and Erik Herve Déjéan. 2001. Introduction to the CoNLL-2001 Shared Task: Clause Identification. In *Proceedings of the Fifth Workshop on Computational Language Learning (CoNLL-2001)*, pages 53–57, Toulouse, France. ACL.

Roser Saurí, Jessica Littman, Robert Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2004. TimeML Annotation Guidelines. http://www.timeml.org.

Andrea Setzer. 2001. *Temporal Information in Newswire Articles: an Annotation Scheme and Corpus Study*. Ph.D. thesis, University of Sheffield, Sheffield, UK.

Marc Verhagen and Robert Knippen. Forthcoming. TANGO: A Graphical Annotation Environment for Ordering Relations. In James Pustejovsky and Robert Gaizauskas, editors, *Time and Event Recognition in Natural Language*. John Benjamin Publications.

Marc Verhagen. 2004. *Times Between The Lines*. Ph.D. thesis, Brandeis University, Waltham, Massachusetts, USA.

Marc Vilain, Henry Kautz, and Peter van Beek. 1990. Constraint propagation algorithms: A revised report. In D. S. Weld and J. de Kleer, editors, *Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufman, San Mateo, California.

# Two diverse systems built using generic components for spoken dialogue (Recent Progress on TRIPS)

**James Allen, George Ferguson, Mary Swift, Amanda Stent, Scott Stoness, Lucian Galescu, Nathan Chambers, Ellen Campana, and Gregory Aist**

University of Rochester
Computer Science Department
UR Comp Sci RC 270226
Rochester NY 14627 USA
{james, ferguson, swift, stoness, campana, gaist}
@cs.rochester.edu

Institute for
Human and Machine Cognition
40 South Alcaniz St.
Pensacola FL 32502
{lgalescu,nchambers}@ihmc.us

State University of New York at
Stony Brook
1418 Computer Science
Stony Brook University
Stony Brook NY 11794 USA
stent@cs.sunysb.edu

## Abstract

This paper describes recent progress on the TRIPS architecture for developing spoken-language dialogue systems. The interactive poster session will include demonstrations of two systems built using TRIPS: a computer purchasing assistant, and an object placement (and manipulation) task.

## 1 Introduction

Building a robust spoken dialogue system for a new task currently requires considerable effort, including extensive data collection, grammar development, and building a dialogue manager that drives the system using its "back-end" application (e.g. database query, planning and scheduling). We describe progress in an effort to build a generic dialogue system that can be rapidly customized to a wide range of different types of applications, primarily by defining a domain-specific task model and the interfaces to the back-end systems. This is achieved by using generic components (i.e., ones that apply in any practical domain) for all stages of understanding and developing techniques for rapidly customizing the generic components to new domains (e.g. Aist, Allen, and Galescu 2004). To achieve this goal we have made several innovations, including (1) developing domain independent models of semantic and contextual interpretation, (2) developing generic dialogue management components based on an abstract model of collaborative problem solving, and (3) extensively using an ontology-mapping system that connects the domain independent representations to the representations/query languages used by the back-end applications, and which is used to automatically optimize the performance of the system in the specific domain.

## 2 Theoretical Underpinnings: The Problem-Solving Model of Dialogue

While many have observed that communication is a specialized form of joint action that happens to involve language and that dialogue can be viewed as collaborative problem solving, very few implemented systems have been explicitly based on these ideas. Theories of speech act interpretation as intention recognition have been developed (including extensive prior work in TRIPS' predecessor, the TRAINS project), but have been generally considered impractical for actual systems. Planning models have been more successful on the generation side, and some systems have used the notion of executing explicit task models to track and drive the interactions (e.g., Sidner and Rich's COLLAGEN framework). But collaborative problem solving, and dialogue in general, is much more general than executing tasks. In our applications, in addition to executing tasks, we see dialogue that is used to define the task (i.e., collaborative planning), evaluate the task (e.g., estimating how long it will take, comparing options, or likely effects), debug a task (e.g., identifying and discussing problems and how to remedy them), learn new tasks (e.g., by demonstration and instruction).

In the remainder of the paper, we'll first discuss the methods we've developed for building dialogue systems using generic components. We'll then describe two systems implemented using the TRIPS architecture that we will demonstrate at the interactive poster session.

## 3 Generic Methods: Ontology Mappings and Collaborative Problem Solving

The goal of our work is to develop generic spoken dialogue technology that can be rapidly customized to new applications, tasks and domains. To do this, we have developed generic domain independent representations not only of sentence meaning but also of the collaborative actions that are performed by the speech acts as one engages in dialogue. Furthermore, we need to be able to easily connect these generic representations to a wide range of different domain specific task models and applications, ranging from data base query systems to state-of-the-art planning and scheduling systems. This paper describes the approach we have developed in the TRIPS system. TRIPS is now being used in a wide range of diverse applications, from interactive planning (e.g., developing evacuation plans), advice giving (e.g., a medication advisor (Ferguson et al. 2002)), controlling teams of robots, collaborative assistance (e.g., an assistant that can help you purchase a computer, as described in this paper), supporting human learning, and most recently having the computer learn (or be taught) tasks, such as learning to perform tasks on the web. Even though the tasks and domains differ dramatically, these applications use the same set of core understanding components.

The key to supporting such a range of tasks and applications is the use of a general ontology-mapping system. This allows the developer to express a set of mapping rules that translate the generic knowledge representation into the specific representations used by the back-end applications (called the KR representation). In order to support generic discourse processing, we represent these mappings as a chain of simpler transformations. These representations are thus transformed in several stages. The first, using the ontology mapping rules, maps the LF representation into an intermediary representation (AKRL - the abstract KR language) that has a generic syntax but whose content is expressed in terms of the KR ontology. The second stage is a syntactic transformation that occurs at the time that calls to the back-end applications actually occur so that interactions occur in the representations the back-end expects. In addition to using ontology mapping to deal with the representational issues, TRIPS is unique in that it uses a generic model of collaborative problem solving to drive the dialogue itself (e.g. Allen, Blaylock, and Ferguson 2002). This model forms the basis of a generic component (the collaboration manager) that supports both intention recognition to identify the intended speech acts and their content, planning the system's actions to respond to the user (or that take initiative), and providing utterance realization goals to the generation system. To develop this, we have been developing a generic ontology of collaborative problem solving acts, which provide the framework for managing the dialogue. The collaboration manager queries a domain-specific task component in order to make decisions about interpretations and responses.

## 4 TRIPS Spoken Dialogue Interface to the CALO Purchasing Assistant

The CALO project is a large multisite effort which aims at building a computerized assistant that learns how to help you with day-to-day tasks. The overarching goal of the CALO project is to

> ... create cognitive software systems, that is, systems that can reason, learn from experience, be told what to do, explain what they are doing, reflect on their experience, and respond robustly to surprise (Mark and Perrault 2004).

Within this broad mandate, one of our current areas of focus is user-system dialogue regarding the task of purchasing - including eliciting user needs, describing possibilities, and reviewing & finalizing a purchase decision. (Not necessarily as discrete stages; these elements may be interleaved as appropriate for the specific item(s) and setting.) Within the purchasing domain, we began with computer purchasing and have branched out to other equipment such as projectors.

How to help with purchasing? The family of tasks involving purchasing items online, regardless of the type of item, have a number of elements in common. The process of purchasing has some common dialogue elements - reporting on the range of features available, allowing the user to specify constraints, and so forth. Also, regarding the goal that must be reached at the end of the task, the eventual item must:

*Meet requirements*. The item needs to meet some sort of user expectations. This could be as arbitrary as a specific part number, or as compositional - and amenable to machine understanding - as a set of physical dimensions (length, width, height, mass, etc.)

*Be approved.* Either the system will have the authority to approve it (cf. Amazon's one-click ordering system), or more commonly the user will review and confirm the purchase. In an office environment the approval process may extend to include review by a supervisor, such as might happen with an item costing over (say) $1000.

*Be available.* (At one time a certain electronics store in California had the habit of leaving out floor models of laptops beyond the point where any were actually available for sale. (Perhaps to entice the unwitting customer into an “upsale”, that is, buying a similar but more expensive computer.)) On a more serious note, computer specifications change rapidly, and so access to online information about available computers (provided by other research within CALO) would be important in order to ensure that the user can actually order the machine he or she has indicated a preference for.

At the interactive poster session, we will demonstrate some of the current spoken dialogue capability related to the CALO task of purchasing equipment. We will demonstrate a number of the aspects of the system such as initiating a conversation, discussing specific requirements, presenting possible equipment to purchase, system-initiated reminders to ask for supervisor approval for large purchases, and finalizing a decision to purchase.

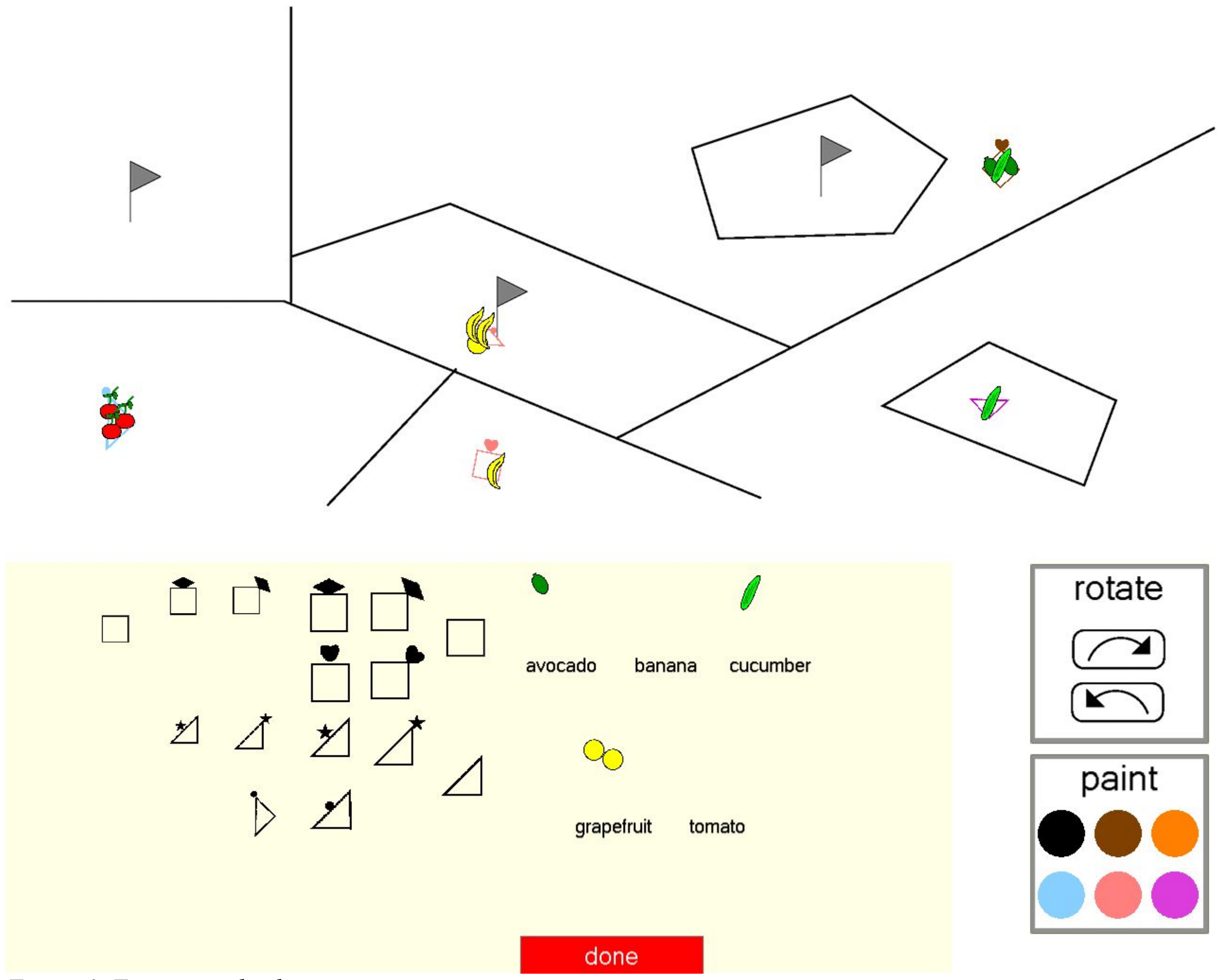


*Figure 1. Fruit carts display.*

## 5 TRIPS Spoken Dialogue Interface to choosing, placing, painting, rotating, and filling (virtual) fruit carts

TRIPS is versatile in its applications, as we've said previously. We hope to also demonstrate an interface to a system for using spoken commands to modifying, manipulating, and placing objects on a computer-displayed map. This system (aka "fruit carts") extends the TRIPS architecture into the realm of continuous understanding. That is, when state-of-the-art dialogue systems listen, they typically wait for the end of the utterance before deciding what to do. People on the other hand do not wait in this way – they can act on partial information as it becomes available. A classic example comes from M. Tanenhaus and colleagues at Rochester: when presented with several objects of various colors and told to "click on the yel-", people will already tend to be looking relatively more at the yellow object(s) even before the word "yellow" has been completed. To achieve this type of interactivity with a dialogue system – at least at the level of two or three words at a time, if not parts of words – imposes some interesting challenges. For example:

1. Information must flow asynchronously between dialogue components, so that actions can be triggered based on partial utterances even while the understanding continues
2. There must be reasonable representations of incomplete information – not just "incomplete sentence", but specifying what is present already and perhaps what may potentially follow
3. Speech recognition, utterance segmentation, parsing, interpretation, discourse reasoning, and actions must all be able to happen in real time

The fruit carts system consists of two main components: first, a graphical interface implemented on Windows 2000 using the .NET framework, and connected to a high-quality eyetracker; second, a TRIPS-driven spoken dialogue interface implemented primarily in LISP. The actions in this domain are as follows:

1. Select an object ("take the large plain square")
2. Move it ("move it to central park")
3. Rotate it ("and then turn it left a bit – that's good")
4. Paint it ("and that one needs to be purple")
5. Fill it ("and there's a grapefruit inside it")

Figure 1 shows an example screenshot from the fruit carts visual display. The natural language interaction is designed to handle various ways of speaking, including conventional definite descriptions ("move the large square to central park") and more interactive language such as ("up towards the flag pole – right a bit – more – um- stop there.")

## 6 Conclusion

In this brief paper, we have described some of the recent progress on the TRIPS platform. In particular we have focused on two systems developed in TRIPS: a spoken dialogue interface to a mixed-initiative purchasing assistant, and a spoken interface for exploring continuous understanding in an object-placement task. In both cases the systems make use of reusable components – for input and output such as parsing and speech synthesis, and also for dialogue functionality such as mapping between language, abstract semantics, and specific representations for each domain.

## References

Aist, G. 2004. Speech, gaze, and mouse data from choosing, placing, painting, rotating, and filling (virtual) vending carts. International Committee for Co-ordination and Standardisation of Speech Databases (COCOSDA) 2004 Workshop, Jeju Island, Korea, October 4, 2004.

Aist, G.S., Allen, J., and Galescu, L. 2004. Expanding the linguistic coverage of a spoken dialogue system by mining human-human dialogue for new sentences with familiar meanings. Member Abstract, 26th Annual Meeting of the Cognitive Science Society, Chicago, August 5-7, 2004.

James Allen, Nate Blaylock, and George Ferguson. A problem-solving model for collaborative agents. In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy, July 15-19 2002.

George Ferguson, James F. Allen, Nate J. Blaylock, Donna K. Byron, Nate W. Chambers, Myrsolava O. Dzikovska, Lucian Galescu, Xipeng Shen, Robert S. Swier, and Mary D. Swift. *The Medication Advisor Project: Preliminary Report*, Technical Report 776, Computer Science Dept., University of Rochester, May 2002.

Mark, B., and Perrault, R. (principal investigators). 2004. Website for Cognitive Assistant that Learns and Organizes. http://www.ai.sri.com/project/CALO

# Transonics: A Practical Speech-to-Speech Translator for English-Farsi Medical Dialogues

**Emil Ettelaie, Sudeep Gandhe, Panayiotis Georgiou, Kevin Knight, Daniel Marcu, Shrikanth Narayanan , David Traum**
University of Southern California
Los Angeles, CA 90089
ettelaie@isi.edu, gandhe@ict.usc.edu, georgiou@sipi.usc.edu, knight@isi.edu, marcu@isi.edu, shri@sipi.usc.edu, traum@ict.use.edu

**Robert Belvin**
HRL Laboratories, LLC
3011 Malibu Canyon Rd.
Malibu, CA 90265
rsbelvin@hrl.com

## Abstract

We briefly describe a two-way speech-to-speech English-Farsi translation system prototype developed for use in doctor-patient interactions. The overarching philosophy of the developers has been to create a system that enables effective communication, rather than focusing on maximizing component-level performance. The discussion focuses on the general approach and evaluation of the system by an independent government evaluation team.

## 1 Introduction

In this paper we give a brief description of a two-way speech-to-speech translation system, which was created under a collaborative effort between three organizations within USC (the Speech Analysis and Interpretation Lab of the Electrical Engineering department, the Information Sciences Institute, and the Institute for Creative Technologies) and the Information Sciences Lab of HRL Laboratories. The system is intended to provide a means of enabling communication between monolingual English speakers and monolingual Farsi (Persian) speakers. The system is targeted at a domain which may be roughly characterized as "urgent care" medical interactions, where the English speaker is a medical professional and the Farsi speaker is the patient. In addition to providing a brief description of the system (and pointers to papers which contain more detailed information), we give an overview of the major system evaluation activities.

## 2 General Design of the system

Our system is comprised of seven speech and language processing components, as shown in Fig. 1. Modules communicate using a centralized message-passing system. The individual subsystems are the Automatic Speech Recognition (ASR) subsystem, which uses n-gram Language Models (LM) and produces n-best lists/lattices along with the decoding confidence scores. The output of the ASR is sent to the Dialog Manager (DM), which displays the n-best and passes one hypothesis on to the translation modules, according to a user-configurable state. The DM sends translation requests to the Machine Translation (MT) unit. The MT unit works in two modes: Classifier based MT and a fully Stochastic MT. Depending on the dialogue manager mode, translations can be sent to the unit selection based Text-To-Speech synthesizer (TTS), to provide the spoken output. The same basic pipeline works in both directions: English ASR, English-Persian MT, Persian TTS, or Persian ASR, Persian-English MT, English TTS.

There is, however, an asymmetry in the dialogue management and control, given the desire for the English-speaking doctor to be in control of the device and the primary "director" of the dialog.

The English ASR used the University of Colorado *Sonic* recognizer, augmented primarily with LM data collected from multiple sources, including

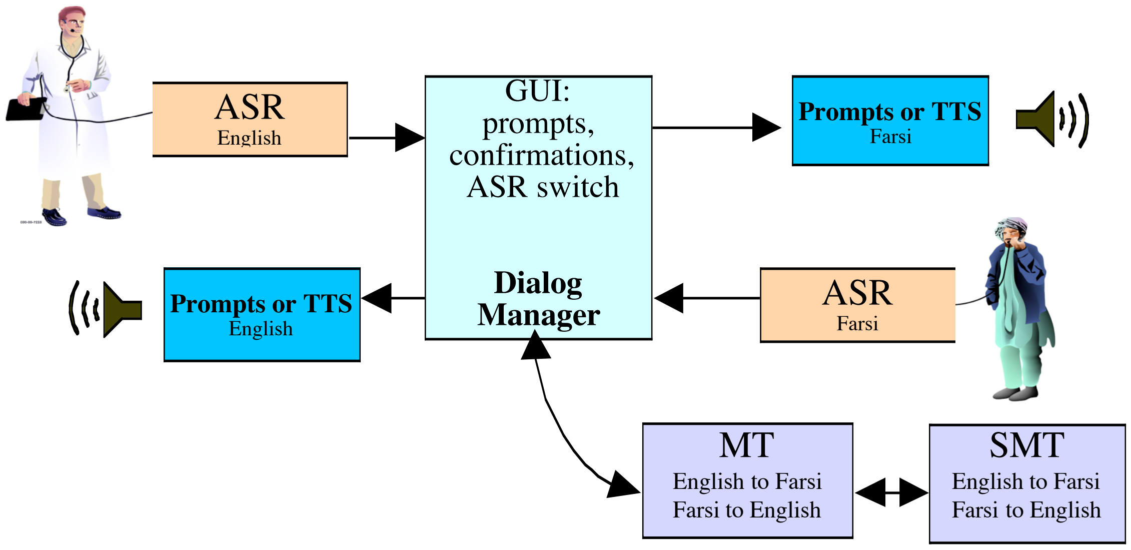


*Figure 1: Architecture of the Transonics system. The Dialogue Manager acts as the hub through which the individual components interact.*

our own large-scale simulated doctor-patient dialogue corpus based on recordings of medical students examining standardized patients (details in Belvin et al. 2004).[1] The Farsi acoustic models r equired an eclectic approach due to the lack of existing labeled speech corpora. The approach included borrowing acoustic data from English by means of developing a sub-phonetic mapping between the two languages, as detailed in (Srinivasamurthy & Narayanan 2003), as well as use of a small existing Farsi speech corpus (FARSDAT), and our own team-internally generated acoustic data. Language modeling data was also obtained from multiple sources. The Defense Language Institute translated approximately 600,000 words of English medical dialogue data (including our standardized patient data mentioned above), and in addition, we were able to obtain usable Farsi text from mining the web for electronic news sources. Other smaller amounts of training data were ob tained from various sources, as detailed in (Narayanan et al. 2003, 2004). Additional detail on development methods for all of these components, system integration and evaluation can also be found in the papers just cited.

The MT components, as noted, consist of both a Classifier and a stochastic translation engine, both developed by USC-ISI team members. The English Classifier uses approximately 1400 classes consisting mostly of standard questions used by medical care providers in medical interviews. Each class has a large number of paraphrases associated with it, such that if the care provider speaks one of those phrases, the system will identify it with the class and translate it to Farsi via table-lookup. If the Classifier cannot succeed in finding a match exceeding a confidence threshold, the stochastic MT engine will be employed. The stochastic MT engine relies on n-gram correspondences between the source and target languages. As with ASR, the performance of the component is highly dependent on very large amounts of training data. Again, there were multiple sources of training data used, the most significant being the data generated by our own team's English collection effort, supported by translation into Farsi by DLI. Further details of the MT components can be found in Narayanan et al., *op.cit*.

## 3 Enabling Effective Communication

The approach taken in the development of Transonics was what can be referred to as the *total communication pathway*. We are not so concerned with trying to maximize the performance of a given component of the system, but rather with the effectiveness of the system as a whole in facilitating actual communication. To this end, our design and development included the following:

[1] *Standardized Patients* are typically actors who have been trained by doctors or nurses to portray symptoms of particular illnesses or injuries. They are used extensively in medical education so that doctors in training don't have to "practice" on real patients.

i. an "educated guess" capability (system guessing at the meaning of an utterance) from the Classifier translation mechanism—this proved very useful for noisy ASR output, especially for the restricted domain of medical interviews.

ii. a flexible and robust SMT good for filling in where the more accurate Classifier misses.

iii. exploitation of a partial n-best list as part of the GUI used by the doctor/medic for the English ASR component and the Farsi-to-English translation component.

iv. a dialog manager which in essence occasionally makes "suggestions" (for next questions for the doctor to ask) based on query sets which are topically related to the query the system believes it recognized the doctor to have spoken.

Overall, the system achieves a respectable level of performance in terms of allowing users to follow a conversational thread in a fairly coherent way, despite the presence of frequent ungrammatical or awkward translations (i.e. despite what we might call *non-catastrophic* errors).

## 4 Testing and Evaluation

In addition to our own laboratory tests, the system was evaluated by MITRE as part of the DARPA program. There were two parts to the MITRE evaluations, a "live" part, designed primarily to evaluate the overall task-oriented effectiveness of the systems, and a "canned" part, designed primarily to evaluate individual components of the systems.

The live evaluation consisted of six medical professionals (doctors, corpsmen and physician's assistants from the Naval Medical Center at Quantico, and a nurse from a civilian institution) conducting unrehearsed "focused history and physical exam" style interactions with Farsi speakers playing the role of patients, where the English-speaking doctor and the Farsi-speaking patient communicated by means of the Transonics system. Since the cases were common enough to be within the realm of general internal medicine, there was no attempt to align ailments with medical specializations among the medical professionals.

MITRE endeavored to find primarily monolingual Farsi speakers to play the role of patient, so as to provide a true test of the system to enable communication between people who would otherwise have no way to communicate. This goal was only partially realized, since one of the two Farsi patient role-players was partially competent in English.[2] The Farsi-speaking role-players were trained by a medical education specialist in how to simulate symptoms of someone with particular injuries or illnesses. Each Farsi-speaking patient role-player received approximately 30 minutes of training for any given illness or injury. The approach was *similar* to that used in training standardized patients, mentioned above (footnote 1) in connection with generation of the dialogue corpus.

MITRE established a number of their own metrics for measuring the success of the systems, as well as using previously established metrics. A full discussion of these metrics and the results obtained for the Transonics system is beyond the scope of this paper, though we will note that one of the most important of these was task-completion. There were 5 significant facts (5 distinct facts for each of 12 different scenarios) that the medical professional should have discovered in the process of interviewing/examining each Farsi patient. The USC/HRL system averaged 3 out of the 5 facts, which was a slightly above-average score among the 4 systems evaluated. A "significant fact" consisted of determining a fact which was critical for diagnosis, such as the fact that the patient had been injured in a fall down a stairway, the fact that the patient was experiencing blurred vision, and so on. Significant facts did not include items such as a patient's age or marital status.[3] We report on this measure in that it is perhaps the single most important component in the assessment, in our opinion, in that it is an indication of many aspects of the system, including *both* directions of the translation system. That is, the doctor will very likely conclude correct findings only if his/her question is translated correctly to the patient, and also if the patient's answer is translated correctly for the doctor. In a true medical exam, the doctor may have

[2] There were additional difficulties encountered as well, having to do with one of the role-players not adequately grasping the goal of role-playing. This experience highlighted the many challenges inherent in simulating domain-specific spontaneous dialogue.

[3] Unfortunately, there was no baseline evaluation this could be compared to, such as assessing whether any of the critical facts could be determined without the use of the system at all.

other means of determining some critical facts even in the absence of verbal communication, but in the role-playing scenario described, this is very unlikely. Although this measure is admittedly coarse-grained, it simultaneously shows, in a crude sense, that the USC/HRL system compared favorably against the other 3 systems in the evaluation, and also that there is still significant room for improvement in the state of the art.

As noted, MITRE devised a *component* evaluation process also consisting of running 5 scripted dialogs through the systems and then measuring ASR and MT performance. The two primary component measures were a version of BLEU for the MT component (modified slightly to handle the much shorter sentences typical of this kind of dialog) and a standard Word-Error Rate for the ASR output. These scores are shown below.

Table 1: Farsi BLEU Scores

| | IBM BLEU ASR | IBM BLEU TEXT |
|---|---|---|
| **English to Farsi** | 0.2664 | 0.3059 |
| **Farsi to English** | 0.2402 | 0.2935 |

The reason for the two different BLEU scores is that one was calculated based on the ASR component output being translated to the other language, while the other was calculated from human transcribed text being translated to the other language.

Table 2: HRL/USC WER for Farsi and English

| | English | Farsi |
|---|---|---|
| WER | 11.5% | 13.4% |

## 5 Conclusion

In this paper we have given an overview of the design, implementation and evaluation of the Transonics speech-to-speech translation system for narrow domain two-way translation. Although there are still many significant hurdles to be overcome before this kind of technology can be called truly robust, with appropriate training and two cooperative interlocutors, we can now see some degree of genuine communication being enabled. And this is very encouraging indeed.

## 6 Acknowledgements

This work was supported primarily by the DARPA CAST/Babylon program, contract N66001-02-C-6023.

## References

R. Belvin, W. May, S. Narayanan, P. Georgiou, S. Ganjavi. 2004. Creation of a Doctor-Patient Dialogue Corpus Using Standardized Patients. In Proceedings of the *Language Resources and Evaluation Conference* (LREC), Lisbon, Portugal.

S. Ganjavi, P. G. Georgiou, and S. Narayanan. 2003. Ascii based transcription schemes for languages with the Arabic script: The case of Persian. In *Proc. IEEE ASRU*, St. Thomas, U.S. Virgin Islands.

S. Narayanan, S. Ananthakrishnan, R. Belvin, E. Ettelaie, S. Ganjavi, P. Georgiou, C. Hein, S. Kadambe, K. Knight, D. Marcu, H. Neely, N. Srinivasamurthy, D. Traum and D. Wang. 2003. Transonics: A speech to speech system for English-Persian Interactions, *Proc. IEEE ASRU*, St. Thomas, U.S. Virgin Islands.

S. Narayanan, S. Ananthakrishnan, R. Belvin, E. Ettelaie, S. Gandhe, S. Ganjavi, P. G. Georgiou, C. M. Hein, S. Kadambe, K. Knight, D. Marcu, H. E. Neely, N. Srinivasamurthy, D. Traum, and D. Wang. 2004. The Transonics Spoken Dialogue Translator: An aid for English-Persian Doctor-Patient interviews, in *Working Notes of the AAAI Fall symposium on Dialogue Systems for Health Communication*, pp 97--103.

N. Srinivasamurthy, and S. Narayanan. 2003. Language adaptive Persian speech recognition. In proceedings of *Eurospeech* 2003.

# The Wild Thing!

**Kenneth Church** **Bo Thiesson**
Microsoft Research
Redmond, WA, 98052, USA
{church, thiesson}@microsoft.com

## Abstract

Suppose you are on a mobile device with no keyboard (e.g., a cell or PDA). How can you enter text quickly? T9? Graffiti? This demo will show how language modeling can be used to speed up data entry, both in the mobile context, as well as the desktop. The Wild Thing encourages users to use wildcards (*). A language model finds the k-best expansions. Users quickly figure out when they can get away with wildcards. General purpose trigram language models are effective for the general case (unrestricted text), but there are important special cases like searching over popular web queries, where more restricted language models are even more effective.

## 1 Motivation: Phone App

Cell phones and PDAs are everywhere. Users love mobility. What are people doing with their phone? You'd think they would be talking on their phones, but a lot of people are typing. It is considered rude to talk on a cell in certain public places, especially in Europe and Asia. SMS text messaging enables people to communicate, even when they can't talk.

It is bizarre that people are typing on their phones given how painful it is. "Talking on the phone" is a collocation, but "typing on the phone" is not. *Slate* (slate.msn.com/id/2111773) recently ran a story titled: "A Phone You Can Actually Type On" with the lead:

> "If you've tried to zap someone a text message recently, you've probably discovered the huge drawback of typing on your cell phone. Unless you're one of those cyborg Scandinavian teenagers who was born with a Nokia in his hand, pecking out even a simple message is a thumb-twisting chore."

There are great hopes that speech recognition will someday make it unnecessary to type on your phone (for SMS or any other app), but speech recognition won't help with the rudeness issue. If people are typing because they can't talk, then speech recognition is not an option. Fortunately, the speech community has developed powerful language modeling techniques that can help even when speech is not an option.

## 2 K-Best String Matching

Suppose we want to search for MSN using a cell phone. A standard approach would be to type 6 <pause> 777 <pause> 66, where 6 → M, 777 → S and 66 → N. (The pauses are necessary for disambiguation.) Kids these days are pretty good at typing this way, but there has to be a better solution.

T9 (www.t9.com) is an interesting alternative. The user types 676 (for *MSN*). The system uses a (unigram) language model to find the k-best matches. The user selects MSN from this list. Some users love T9, and some don't.

The input, 676, can be thought of as short hand for the regular expression:

```
/^[6MNOmno][7PRSprs][6MNOmno]$/
```

using standard Unix notation. Regular expressions become much more interesting when we consider wildcards. So-called "word wheeling" can be thought of as the special case where we add a wildcard to the end of whatever the user types. Thus, if the user types 676 (for MSN), we would find the k-best matches for:

```
/^[6MNOmno][7PRSprs][6MNOmno].*/
```

See Google Suggests[1] for a nice example of word wheeling. Google Suggests makes it easy to find popular web queries (in the standard non-mobile desktop context). The user types a prefix. After each character, the system produces a list of the *k* most popular web queries that start with the specified prefix.

Word wheeling not only helps when you know what you want to say, but it also helps when you don't. Users can't spell. And things get stuck on the tip of their tongue. Some users are just browsing. They aren't looking for anything in particular, but they'd like to know what others are looking at.

The popular query application is relatively easy in terms of entropy. About 19 bits are needed to specify one of the 7 million most popular web queries. That is, if we assign each web query a probability based on query logs collected at msn.com, then we can estimate entropy, H, and discover that $H \approx 19$. (About 23 bits would be needed if these pages were equally likely, but they aren't.) It is often said that the average query is between two and three words long, but H is more meaningful than query length.

General purpose trigram language models are effective for the general case (unrestricted text), but there are important special cases like popular web queries, where more restricted language models are even more effective than trigram models. Our language model for web queries is simply a list of queries and their probabilities. We consider queries to be a finite language, unlike unrestricted text where the trigram language model allows sentences to be arbitrarily long.

Let's consider another example. The *MSN* query was too easy. Suppose we want to find *Condoleezza Rice*, but we can't spell her name. And even if we could, we wouldn't want to. Typing on a phone isn't fun.

We suggest spelling *Condoleezza* as 2*, where 2 $\rightarrow$ [ABCabc2] and * is the wildcard. We then type '#' for space. Rice is easy to spell: 7423. Thus, the user types, 2*#7423, and the system searches over the MSN query log to produce a list of k-best (most popular) matches (*k* defaults to 10):

1. Anne Rice
2. Book of Shadows
3. Chris Rice
4. Condoleezza Rice
5. Ann Rice

   …

8. Condoleezza Rice

The letters matching constants in the regular expression are underlined. The other letters match wildcards. (An implicit wildcard is appended to the end of the input string.)

Wildcards are very powerful. Strings with wildcards are more expressive than prefix matching (word wheeling). As mentioned above, it should take just 19 bits on average to specify one of the 7 million most popular queries. The query 2*#7423 contains 7 characters in an 12-character alphabet (2-9 $\rightarrow$ [A-Za-z2-9] in the obvious way, except that 0 $\rightarrow$ [QZqz0]; # $\rightarrow$ space; * is wild). 7 characters in a 12 character alphabet is $7 \log_2 12 = 25$ bits. If the input notation were optimal (which it isn't), it shouldn't be necessary to type much more than this on average to specify one of the 7 million most popular queries.

Alphabetic ordering causes bizarre behavior. Yellow Pages are full of company names starting with *A, AA, AAA*, etc.. If prefix matching tools like *Google Suggests* take off, then it is just a matter of time before companies start to go after valuable prefixes: *mail*, *maps*, etc. Wildcards can help society avoid that non-sense. If you want to find a top mail site, you can type, "*mail" and you'll find: *Gmail*, *Hotmail*, *Yahoo mail*, etc..

## 3 Collaboration & Personalization

Users quickly learn when they can get away with wildcards. Typing therefore becomes a collaborative exercise, much like Palm's approach to handwriting recognition. Recognition is hard. Rather than trying to solve the general case, Palm encourages users to work with the system to write in a way that is easier to recognize (Graffiti). The system isn't trying to solve the AI problem by itself, but rather there is a man-machine collaboration where both parties work together as a team.

Collaboration is even more powerful in the web context. Users issue lots of queries, making it clear what's hot (and what's not). The system constructs a language model based on these queries to direct users toward good stuff. More and more users will then go there, causing the hot query to move up in the language model. In this way, collaboration can be viewed as a positive feedback

[1] http://www.google.com/webhp?complete=1

loop. There is a strong herd instinct; all parties benefit from the follow-the-pack collaboration.

In addition, users want personalization. When typing names of our friends and family, technical terms, etc., we should be able to get away with more wildcards than other users would. There are obvious opportunities for personalizing the language model by integrating the language model with a desktop search index (Dumais *et al*, 2003).

## 4 Modes, Language Models and Apps

The Wild Thing demo has a switch for turning on and off phone mode to determine whether input comes from a phone keypad or a standard keyboard. Both with and without phone mode, the system uses a language model to find the k-best expansions of the wildcards.

The demo contains a number of different language models, including a number of standard trigram language models. Some of the language models were trained on large quantities (6 Billion words) of English. Others were trained on large samples of Spanish and German. Still others were trained on small sub-domains (such as ATIS, available from www.ldc.upenn.edu). The demo also contains two special purpose language models for searching popular web queries, and popular web domains.

Different language models are different. With a trigram language model trained on general English (containing large amounts of newswire collected over the last decade),

```
pres* rea* *d y* t* it is v*
imp* → President Reagan said
yesterday that it is very impor-
tant
```

With a Spanish Language Model,

```
pres* rea* → presidente Reagan
```

In the ATIS domain,

```
pres* rea* → <UNK> <UNK>
```

The tool can also be used to debug language models. It turns out that some French slipped into the English training corpus. Consequently, the English language model expanded the * in *en * de* to some common French words that happen to be English words as well: *raison, circulation, oeuvre, place*, as well as <OOV>. After discovering this, we discovered quite a few more anomalies in the training corpus such as headers from the AP news.

There may also be ESL (English as a Second Language) applications for the tool. Many users have a stronger active vocabulary than passive vocabulary. If the user has a word stuck on the tip of their tongue, they can type a suggestive context with appropriate wildcards and there is a good chance the system will propose the word the user is looking for.

Similar tricks are useful in monolingual contexts. Suppose you aren't sure how to spell a celebrity's name. If you provide a suggestive context, the language model is likely to get it right:

```
ron* r*g*n → Ronald Reagan
don* r*g*n → Donald Regan
c* rice → Condoleezza Rice
```

To summarize, wildcards are helpful in quite a few apps:

- No keyboard: cell phone, PDA, Tablet PC.
- Speed matters: instant messaging, email.
- Spelling/ESL/tip of the tongue.
- Browsing: direct users toward hot stuff.

## 5 Indexing and Compression

The k-best string matching problem raises a number of interesting technical challenges. We have two types of language models: trigram language models and long lists (for finite languages such as the 7 million most popular web queries).

The long lists are indexed with a suffix array. Suffix arrays[2] generalize very nicely to phone mode, as described below. We treat the list of web queries as a text of $N$ bytes. (Newlines are replaced with end-of-string delimiters.) The suffix array, $S$, is a sequence of $N$ ints. The array is initialized with the ints from 0 to $N-1$. Thus, $S[i]=i$, for $0 \leq i < N$. Each of these ints represents a string, starting at position $i$ in the text and extending to the end of the string. $S$ is then sorted alphabetically.

Suffix arrays make it easy to find the frequency and location of any substring. For example, given the substring "mail," we find the first and last suffix in $S$ that starts with "mail." The gap between these two is the frequency. Each suffix in the gap points to a super-string of "mail."

To generalize suffix arrays for phone mode we replace alphabetical order (strcmp) with phone order (phone-strcmp). Both strcmp and phone-strcmp consider each character one at a time. In standard alphabetic ordering, 'a'<'b'<'c', but in

[2] An excellent discussion of suffix arrays including source code can be found at www.cs.dartmouth.edu/~doug.

phone-strcmp, the characters that map to the same key on the phone keypad are treated as equivalent.

We generalize suffix arrays to take advantage of popularity weights. We don't want to find all queries that contain the substring "mail," but rather, just the k-best (most popular). The standard suffix array method will work, if we add a filter on the output that searches over the results for the k-best. However, that filter could take O($N$) time if there are lots of matches, as there typically are for short queries.

An improvement is to sort the suffix array by both popularity and alphabetic ordering, alternating on even and odd depths in the tree. At the first level, we sort by the first order and then we sort by the second order and so on, using a construction, vaguely analogous to KD-Trees (Bentley, 1975). When searching a node ordered by alphabetical order, we do what we would do for standard suffix arrays. But when searching a node ordered by popularity, we search the more popular half before the second half. If there are lots of matches, as there are for short strings, the index makes it very easy to find the top-k quickly, and we won't have to search the second half very often. If the prefix is rare, then we might have to search both halves, and therefore, half the splits (those split by popularity) are useless for the worst case, where the input substring doesn't match anything in the table. Lookup is O(sqrt $N$).[3]

Wildcard matching is, of course, a different task from substring matching. Finite State Machines (Mohri *et al*, 2002) are the right way to think about the k-best string matching problem with wildcards. In practice, the input strings often contain long anchors of constants (wildcard free substrings). Suffix arrays can use these anchors to generate a list of candidates that are then filtered by a regex package.

[3] Let $F(N)$ be the work to process $N$ items on the frequency splits and let $A(N)$ be the work to process $N$ items on the alphabetical splits. In the worst case, $F(N) = 2A(N/2) + C_1$ and $A(N) = F(N/2) + C_2$, where $C_1$ and $C_2$ are two constants. In other words, $F(N) = 2F(N/4) + C$, where $C = C_1 + 2C_2$. We guess that $F(N) = \alpha$ sqrt($N$) + $\beta$, where $\alpha$ and $\beta$ are constant. Substituting this guess into the recurrence, the dependencies on $N$ cancel. Thus, we conclude, F($N$) = O(sqrt $N$).

Memory is limited in many practical applications, especially in the mobile context. Much has been written about lossless compression of language models. For trigram models, we use a lossy method inspired by the Unix Spell program (McIlroy, 1982). We map each trigram $<x, y, z>$ into a hash code $h = (V^2 x + V y + z) \% P$, where $V$ is the size of the vocabulary and $P$ is an appropriate prime. $P$ trades off memory for loss. The cost to store $N$ trigrams is: $N$ [$1/\log_e 2 + \log_2(P/N)$] bits. The loss, the probability of a false hit, is $1/P$.

The $N$ trigrams are hashed into $h$ hash codes. The codes are sorted. The differences, $x$, are encoded with a Golomb code[4] (Witten *et al*, 1999), which is an optimal Huffman code, assuming that the differences are exponentially distributed, which they will be, if the hash is Poisson.

## 6 Conclusions

The Wild Thing encourages users to make use of wildcards, speeding up typing, especially on cell phones. Wildcards are useful when you want to find something you can't spell, or something stuck on the tip of your tongue. Wildcards are more expressive than standard prefix matching, great for users, and technically challenging (and fun) for us.

## References

J. L. Bentley (1975), Multidimensional binary search trees used for associative searching, *Commun. ACM*, 18:9, pp 509-517.

S. T. Dumais, E. Cutrell, et al (2003). Stuff I've Seen: A system for personal information retrieval and re-use, *SIGIR*.

M. D. McIlroy (1982), Development of a spelling list, *IEEE Trans. on Communications* **30**, 91-99.

M. Mohri, F. C. N. Pereira, and M. Riley. Weighted Finite-State Transducers in Speech Recognition. *Computer Speech and Language*, 16(1):69-88, 2002.

I. H. Witten, A. Moffat and T. C. Bell, (1999), *Managing Gigabytes: Compressing and Indexing Documents and Images,* by Morgan Kaufmann Publishing, San Francisco, ISBN 1-55860-570-3.

[4] In Golomb, $x = x_q\, m + x_r$, where $x_q$ = floor($x/m$) and $x_r = x$ mod $m$. Choose m to be a power of two near ceil(½ E[$x$])=ceil(½ $P/N$). Store quotients $x_q$ in unary and remainders $x_r$ in binary. $z$ in unary is a sequence of z−1 zeros followed by a 1. Unary is an optimal Huffman code when Pr($z$)=$(½)^{z+1}$. Storage costs are: $x_q$ bits for $x_q$ + $\log_2 m$ bits for $x_r$.

# Interactively Exploring a Machine Translation Model

**Steve DeNeefe, Kevin Knight, and Hayward H. Chan**
Information Sciences Institute and Department of Computer Science
The Viterbi School of Engineering, University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{sdeneefe,knight}@isi.edu, hhchan@umich.edu

**Abstract**

This paper describes a method of interactively visualizing and directing the process of translating a sentence. The method allows a user to explore a model of syntax-based statistical machine translation (MT), to understand the model's strengths and weaknesses, and to compare it to other MT systems. Using this visualization method, we can find and address conceptual and practical problems in an MT system. In our demonstration at ACL, new users of our tool will drive a syntax-based decoder for themselves.

## 1 Introduction

There are many new approaches to statistical machine translation, and more ideas are being suggested all the time. However, it is difficult to determine how well a model will actually perform. Experienced researchers have been surprised by the capability of unintuitive word-for-word models; at the same time, seemingly capable models often have serious hidden problems — intuition is no substitute for experimentation. With translation ideas growing more complex, capturing aspects of linguistic structure in different ways, it becomes difficult to try out a new idea without a large-scale software development effort.

Anyone who builds a full-scale, trainable translation system using syntactic information faces this problem. We know that syntactic models often do not fit the data. For example, the syntactic system described in Yamada and Knight (2001) cannot translate n-to-m-word phrases and does not allow for multi-level syntactic transformations; both phenomena are frequently observed in real data. In building a new syntax-based MT system which addresses these flaws, we wanted to find problems in our framework as early as possible. So we decided to create a tool that could help us answer questions like:

1. Does our framework allow good translations for real data, and if not, where does it get stuck?
2. How does our framework compare to existing state-of-the-art phrase-based statistical MT systems such as Och and Ney (2004)?

The result is DerivTool, an interactive translation visualization tool. It allows a user to build up a translation from one language to another, step by step, presenting the user with the myriad of choices available to the decoder at each point in the process. DerivTool simplifies the user's experience of exploring these choices by presenting only the decisions relevant to the context in which the user is working, and allowing the user to search for choices that fit a particular set of conditions. Some previous tools have allowed the user to visualize word alignment information (Callison-Burch et al., 2004; Smith and Jahr, 2000), but there has been no corresponding deep effort into visualizing the decoding experience itself. Other tools use visualization to aid the user in manually developing a grammar (Copestake and Flickinger, 2000), while our tool visualizes

| | |
|---|---|
| Starting with: | 被 警方 击毙 |
| and applying the rule: | `NPB(DT(the) NNS(police))` ↔ 警方 |
| we get: | 被 `NPB(DT(the) NNS(police))` 击毙 |
| If we then apply the rule: | `VBN(killed)` ↔ 击毙 |
| we get: | 被 `NPB(DT(the) NNS(police)) VBN(killed)` |
| Applying the next rule: | `NP-C(x0:NPB)` ↔ `x0` |
| results in: | 被 `NP-C(NPB(DT(the) NNS(police))) VBN(killed)` |
| Finally, applying the rule: | `VP(VBD(was) VP-C(x0:VBN PP(IN(by) x1:NP-C)))` ↔ 被 `x1 x0` |
| results in the final phrase: | `VP(VBD(was) VP-C(VBN(killed) PP(IN(by) NP-C(NPB(DT(the) NNS(police))))))` |

Table 1: By applying applying four rules, a Chinese verb phrase is translated to English.

the translation process itself, using rules from very large, automatically learned rule sets. DerivTool can be adapted to visualize other syntax-based MT models, other tree-to-tree or tree-to-string MT models, or models for paraphrasing.

## 2 Translation Framework

It is useful at this point to give a brief description of the syntax-based framework that we work with, which is based on translating Chinese sentences into English syntax trees. Galley et al. (2004) describe how to learn hundreds of millions of tree-transformation rules from a parsed, aligned Chinese/English corpus, and Galley et al. (submitted) describe probability estimators for those rules. We decode a new Chinese sentence with a method similar to parsing, where we apply learned rules to build up a complete English tree hypothesis from the Chinese string.

The rule extractor learns rules for many situations. Some are simple phrase-to-phrase rules such as:

`NPB(DT(the) NNS(police))` ↔ 警方

This rule should be read as follows: replace the Chinese word 警方 with the noun phrase "the police". Others rules can take existing tree fragments and build upon them. For example, the rule

`S(x0:NP-C x1:VP x2:.)` ↔ `x0 x1 x2`

takes three parts of a sentence, a noun phrase (x0), a verb phrase (x1), and a period (x2) and ties them together to build a complete sentence. Rules also can involve phrase re-ordering, as in

`NPB(x0:JJ x1:NN)` ↔ `x1 x0`

This rule builds an English noun phrase out of an adjective (x0) and a noun (x1), but in the Chinese, the order is reversed. Multilevel rules can tie several of these concepts together; the rule

`VP(VBD(was) VP-C(x0:VBN PP(IN(by) x1:NP-C)))`
↔ 被 `x1 x0`

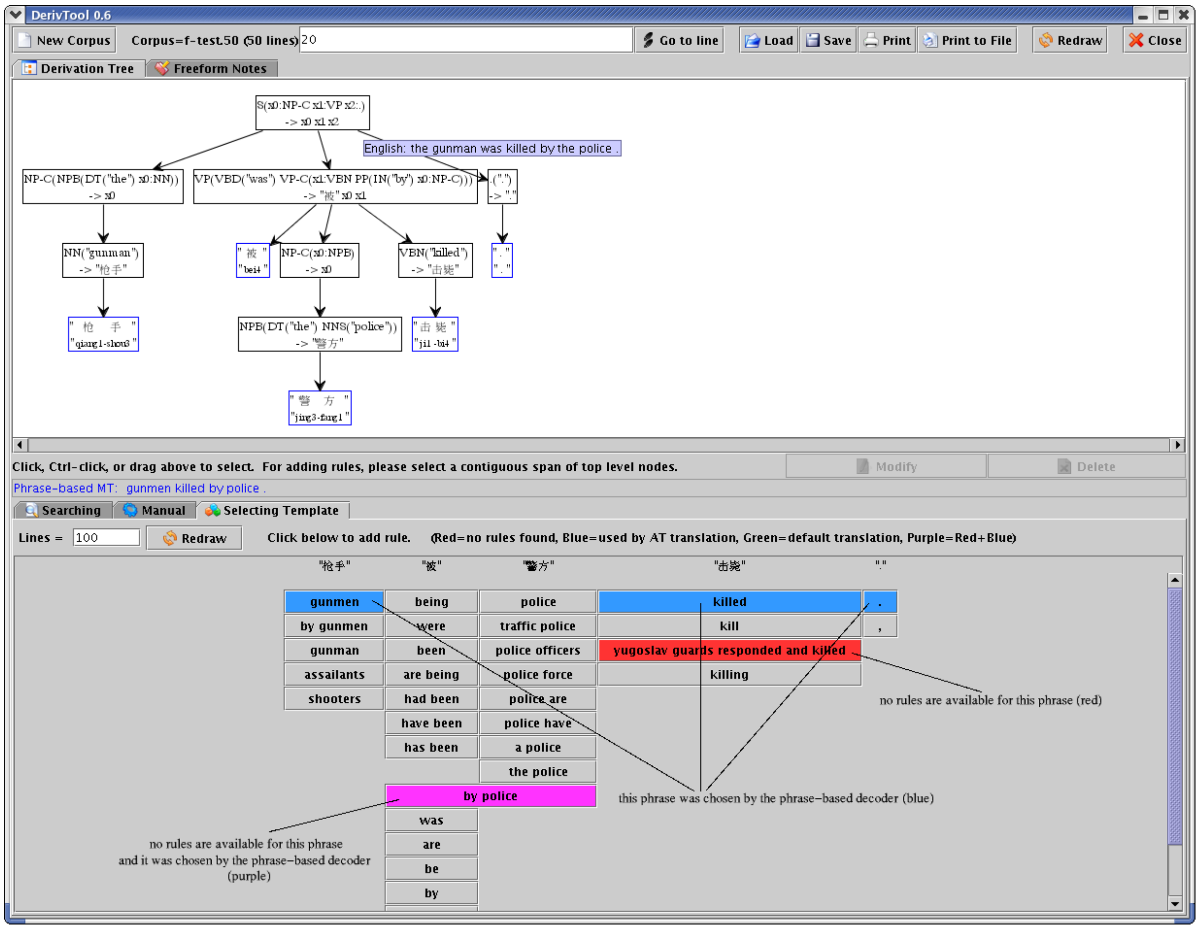
takes a Chinese word 被 and two English constituents — x1, a noun phrase, and x0, a past-participle verb — and translates them into a phrase of the form "was [verb] by [noun-phrase]". Notice that the order of the constituents has been reversed in the resulting English phrase, and that English function words have been generated.

The decoder builds up a translation from the Chinese sentence into an English tree by applying these rules. It follows the decoding-as-parsing idea exemplified by Wu (1996) and Yamada and Knight (2002). For example, the Chinese verb phrase 被 警方 击毙 (literally, "[passive] police kill") can be translated to English via four rules (see Table 1).

## 3 DerivTool

In order to test whether good translations can be generated with rules learned by Galley et al. (2004), we created DerivTool as an environment for interactively using these rules as a decoder would. A user starts with a Chinese sentence and applies rules one after another, building up a translation from Chinese to English. After finishing the translation, the user can save the trace of rule-applications (the *derivation tree*) for later analysis.

We now outline the typical procedure for a user to translate a sentence with DerivTool. To start, the user loads a set of sentences to translate and chooses a particular one to work with. The tool then presents the user with a window split halfway up. The top



English: the gunman was killed by the police .

Click, Ctrl-click, or drag above to select. For adding rules, please select a contiguous span of top level nodes.

Phrase-based MT: gunmen killed by police .

Click below to add rule. (Red=no rules found, Blue=used by AT translation, Green=default translation, Purple=Red+Blue)

| "枪手" | "被" | "警方" | "击毙" | "." |
|---|---|---|---|---|
| gunmen | being | police | killed | . |
| by gunmen | were | traffic police | kill | , |
| gunman | been | police officers | yugoslav guards responded and killed | |
| assailants | are being | police force | killing | |
| shooters | had been | police are | | |
| | have been | police have | | |
| | has been | a police | | |
| | | the police | | |
| | by police | by police | | |
| | was | | | |
| | are | | | |
| | be | | | |
| | by | | | |

no rules are available for this phrase (red)

this phrase was chosen by the phrase–based decoder (blue)

no rules are available for this phrase and it was chosen by the phrase–based decoder (purple)

Figure 1: DerivTool with a completed derivation.

half is the workspace where the user builds a translation. It initially displays only the Chinese sentence, with each word as a separate node. The bottom half presents a set of tabbed panels which allow the user to select rules to build up the translation. See Figure 1 for a picture of the interface showing a completed derivation tree.

The most immediately useful panel is called *Selecting Template*, which shows a grid of possible English phrasal translations for Chinese phrases from the sentence. This phrase grid contains both phrases learned in our extracted rules (e.g., "the police" from earlier) and phrases learned by the phrase-based translation system (Och and Ney, 2004)[1]. The user presses a grid button to choose a phrase to include in the translation. At this point, a frequency-ordered list of rules will appear; these rules translate the Chinese phrase into the button-selected English phrase, and the user specifies which one to use. Often there will be more than one rule (e.g., 击毙 may translate via the rule `VBD(killed)` ↔ 击毙 or `VBN(killed)` ↔ 击毙), and sometimes there are no rules available. When there are no rules, the buttons are marked in red, telling us that the phrase-based system has access to this phrasal translation but our learned syntactic rules did not capture it. Other buttons are marked green to represent translations from the specialized number/name/date system, and others are blue, indicating the phrases in the phrase-based decoder's best output. A purple button indicates both red and blue, i.e., the phrase was chosen by the phrase-based decoder but is unavailable in our syntactic framework. This is a bad combination, showing us where rule learning is weak. The

[1]The phrase-based system serves as a sparring partner. We display its best decoding in the center of the screen. Note that in Figure 1 its output lacks an auxiliary verb and an article.

remaining buttons are gray.

Once the user has chosen the phrasal rules required for translating the sentence, the next step is to stitch these phrases together into a complete English syntax tree using more general rules. These are found in another panel called *Searching*. This panel allows a user to select a set of adjacent, top-level nodes in the tree and find a rule that will connect them together. It is commonly used for building up larger constituents from smaller ones. For example, if one has a noun-phrase, a verb-phrase, and a period, the user can search for the rule that connects them and builds an "S" on top, completing the sentence. The results of a search are presented in a list, again ordered by frequency.

A few more features to note are: 1) loading and saving your work at any point, 2) adding free-form notes to the document (e.g. "I couldn't find a rule that..."), and 3) manually typing rules if one cannot be found by the above methods. This allows us to see deficiencies in the framework.

## 4 How DerivTool Helps

First, DerivTool has given us confidence that our syntax-based framework can work, and that the rules we are learning are good. We have been able to manually build a good translation for each sentence we tried, both for short and long sentences. In fact, there are multiple good ways to translate sentences using these rules, because different DerivTool users translate sentences differently. Ordering rules by frequency and/or probability helps us determine if the rules we want are also frequent and favored by our model.

DerivTool has also helped us to find problems with the framework and to see clearly how to fix them. For example, in one of our first sentences we realized that there was no rule for translating a date — likewise for numbers, names, currency values, and times of day. Our phrase-based system solves these problems with a specialized date/name/number translator. Through the process of manually typing syntactic transformation rules for dates and numbers in DerivTool, it became clear that our current date/name/number translator did not provide enough information to create such syntactic rules automatically. This sparked a new area of research before we had a fully-functional decoder.

We also found that multi-word noun phrases, such as "Israeli Prime Minister Sharon" and "the French Ambassador's visit" were often parsed in a way that did not allow us to learn good translation rules. The flat structure of the constituents in the syntax tree makes it difficult to learn rules that are general enough to be useful. Phrases with possessives also gave particular difficulty due to the awkward multilevel structure of the parser's output. We are researching solutions to these problems involving restructuring the syntax trees before training.

Finally, our tool has helped us find bugs in our system. We found many cases where rules we wanted to use were unexpectedly absent. We eventually traced these bugs to our rule extraction system. Our decoder would have simply worked around this problem, producing less desirable translations, but DerivTool allowed us to quickly spot the missing rules.

## 5 Conclusion

We created DerivTool to test our MT framework against real-world data before building a fully-functional decoder. By allowing us to play the role of a decoder and translate sentences manually, it has given us insight into how well our framework fits the data, what some of its weaknesses are, and how it compares to other systems. We continue to use it as we try out new rule-extraction techniques and finish the decoding system.

## References

Chris Callison-Burch, Colin Bannard and Josh Schroeder. 2004. Improved statistical translation through editing. *EAMT-2004 Workshop.*

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. *Proc. of LREC 2000.*

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? *Proc. of NAACL-HLT 2004.*

Franz Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4).

Noah A. Smith and Michael E. Jahr. 2000. Cairo: An Alignment Visualization Tool. *Proc. of LREC 2000.*

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. *Proc. of ACL.*

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. *Proc. of ACL.*

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. *Proc. of ACL.*

# Multi-Engine Machine Translation Guided by Explicit Word Matching

**Shyamsundar Jayaraman**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
shyamj@cs.cmu.edu

**Alon Lavie**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
alavie@cs.cmu.edu

## Abstract

We describe a new approach for synthetically combining the output of several different Machine Translation (MT) engines operating on the same input. The goal is to produce a synthetic combination that surpasses all of the original systems in translation quality. Our approach uses the individual MT engines as "black boxes" and does not require any explicit cooperation from the original MT systems. A decoding algorithm uses explicit word matches, in conjunction with confidence estimates for the various engines and a trigram language model in order to score and rank a collection of sentence hypotheses that are synthetic combinations of words from the various original engines. The highest scoring sentence hypothesis is selected as the final output of our system. Experiments, using several Arabic-to-English systems of similar quality, show a substantial improvement in the quality of the translation output.

## 1 Introduction

A variety of different paradigms for machine translation (MT) have been developed over the years, ranging from statistical systems that learn mappings between words and phrases in the source language and their corresponding translations in the target language, to Interlingua-based systems that perform deep semantic analysis. Each approach and system has different advantages and disadvantages. While statistical systems provide broad coverage with little manpower, the quality of the corpus based systems rarely reaches the quality of knowledge based systems.

With such a wide range of approaches to machine translation, it would be beneficial to have an effective framework for combining these systems into an MT system that carries many of the advantages of the individual systems and suffers from few of their disadvantages. Attempts at combining the output of different systems have proved useful in other areas of language technologies, such as the ROVER approach for speech recognition (Fiscus 1997). Several approaches to multi-engine machine translation systems have been proposed over the past decade. The Pangloss system and work by several other researchers attempted to combine lattices from many different MT systems (Frederking et Nirenburg 1994, Frederking et al 1997; Tidhar & Küssner 2000; Lavie, Probst et al. 2004). These systems suffer from requiring cooperation from all the systems to produce compatible lattices as well as the hard research problem of standardizing confidence scores that come from the individual engines. In 2001, Bangalore et al used string alignments between the different translations to train a finite state machine to produce a *consensus* translation. The alignment algorithm described in that work, which only allows insertions, deletions and substitutions, does not accurately capture long range phrase movement.

In this paper, we propose a new way of combining the translations of multiple MT systems based on a more versatile word alignment algorithm. A "decoding" algorithm then uses these alignments, in conjunction with confidence estimates for the various engines and a trigram language model, in order to score and rank a collection of sentence hypotheses that are synthetic combinations of words from the various original engines. The highest scoring sentence hypothesis is selected as the final output of our system. We experimentally tested the new approach by combining translations obtained from combining three Arabic-to-English translation systems. Translation quality is scored using the METEOR MT evaluation metric (Lavie, Sagae et al 2004). Our experiments demonstrate that our new MEMT system achieves a substantial improvement over all of the original systems, and also outperforms an “oracle” capable of selecting the best of the original systems on a sentence-by-sentence basis.

The remainder of this paper is organized as follows. In section 2 we describe the algorithm for generating multi-engine synthetic translations. Section 3 describes the experimental setup used to evaluate our approach, and section 4 presents the results of the evaluation. Our conclusions and directions for future work are presented in section 5.

## 2 The MEMT Algorithm

Our Multi-Engine Machine Translation (MEMT) system operates on the single “top-best” translation output produced by each of several MT systems operating on a common input sentence. MEMT first aligns the words of the different translation systems using a word alignment matcher. Then, using the alignments provided by the matcher, the system generates a set of synthetic sentence hypothesis translations. Each hypothesis translation is assigned a score based on the alignment information, the confidence of the individual systems, and a language model. The hypothesis translation with the best score is selected as the final output of the MEMT combination.

### 2.1 The Word Alignment Matcher

The task of the matcher is to produce a word-to-word alignment between the words of two given input strings. Identical words that appear in both input sentences are potential matches. Since the same word may appear multiple times in the sentence, there are multiple ways to produce an alignment between the two input strings. The goal is to find the alignment that represents the best correspondence between the strings. This alignment is defined as the alignment that has the smallest number of “crossing edges. The matcher can also consider morphological variants of the same word as potential matches. To simultaneously align more than two sentences, the matcher simply produces alignments for all pair-wise combinations of the set of sentences.

In the context of its use within our MEMT approach, the word-alignment matcher provides three main benefits. First, it explicitly identifies translated words that appear in multiple MT translations, allowing the MEMT algorithm to reinforce words that are common among the systems. Second, the alignment information allows the algorithm to ensure that aligned words are not included in a synthetic combination more than once. Third, by allowing long range matches, the synthetic combination generation algorithm can consider different plausible orderings of the matched words, based on their location in the original translations.

### 2.2 Basic Hypothesis Generation

After the matcher has word aligned the original system translations, the decoder goes to work. The hypothesis generator produces synthetic combinations of words and phrases from the original translations that satisfy a set of adequacy constraints. The generation algorithm is an iterative process and produces these translation hypotheses incrementally. In each iteration, the set of existing partial hypotheses is extended by incorporating an additional word from one of the original translations. For each partial hypothesis, a data-structure keeps track of the words from the original translations which are accounted for by this partial hypothesis. One underlying constraint observed by the generator is that the original translations are considered in principle to be word synchronous in the sense that selecting a word from one original translation normally implies “marking” a corresponding word in each of the other original translations as “used”. The way this is determined is explained below. Two partial hypotheses that have the same partial translation, but have a different set of words that have been accounted for are considered different. A hypothesis is considered “complete” if the next word chosen to extend the hypothesis is the explicit end-of-sentence marker from one of the original translation strings. At the start of hypothesis generation, there is a single hypothesis, which has the empty string as its partial translation and where none of the words in any of the original translations are marked as used.

In each iteration, the decoder extends a hypothesis by choosing the *next* unused word from one of the original translations. When the decoder chooses to extend a hypothesis by selecting word *w* from original system A, the decoder marks *w* as *used*. The decoder then proceeds to identify and mark as used a word in each of the other original systems. If *w* is aligned to words in any of the other original translation systems, then the words that are aligned with *w* are also marked as used. For each system that does not have a word that aligns with *w*, the decoder establishes an *artificial alignment* between *w* and a word in this system. The intuition here is that this artificial alignment corresponds to a different translation of the same source-language word that corresponds to *w*. The choice of an artificial alignment cannot violate constraints that are imposed by alignments that were found by the matcher. If no artificial alignment can be established, then no word from this system will be marked as used. The decoder repeats this process for each of the original translations. Since the order in which the systems are processed matters, the decoder produces a separate hypothesis for each order.

Each iteration expands the previous set of partial hypotheses, resulting in a large space of complete synthetic hypotheses. Since this space can grow exponentially, pruning based on scoring of the partial hypotheses is applied when necessary.

### 2.3 Confidence Scores

A major component in the scoring of hypothesis translations is a confidence score that is assigned to each of the original translations, which reflects the translation adequacy of the system that produced it. We associate a confidence score with each word in a synthetic translation based on the confidence of the system from which it originated. If the word was contributed by several different original translations, we sum the confidences of the contributing systems. This word confidence score is combined multiplicatively with a score assigned to the word by a trigram language model. The score assigned to a complete hypothesis is its geometric average word score. This removes the inherent bias for shorter hypotheses that is present in multiplicative cumulative scores.

### 2.4 Restrictions on Artificial Alignments

The basic algorithm works well as long the original translations are reasonably word synchronous. This rarely occurs, so several additional constraints are applied during hypothesis generation. First, the decoder discards unused words in original systems that "linger" around too long. Second, the decoder limits how far ahead it looks for an artificial alignment, to prevent incorrect long-range artificial alignments. Finally, the decoder does not allow an artificial match between words that do not share the same part-of-speech.

## 3 Experimental Setup

We combined outputs of three Arabic-to-English machine translation systems on the 2003 TIDES Arabic test set. The systems were AppTek's rule based system, CMU's EBMT system, and Systran's web-based translation system.

We compare the results of MEMT to the individual online machine translation systems. We also compare the performance of MEMT to the score of an "oracle system" that chooses the best scoring of the individual systems for each sentence. Note that this oracle is not a realistic system, since a real system cannot determine at runtime which of the original systems is best on a sentence-by-sentence basis. One goal of the evaluation was to see how rich the space of synthetic translations produced by our hypothesis generator is. To this end, we also compare the output selected by our current MEMT system to an "oracle system" that chooses the best synthetic translation that was generated by the decoder for each sentence. This too is not a realistic system, but it allows us to see how well our hypothesis scoring currently performs. This also provides a way of estimating a performance ceiling of the MEMT approach, since our MEMT can only produce words that are provided by the original systems (Hogan and Frederking 1998).

Due to the computational complexity of running the oracle system, several practical restrictions were imposed. First, the oracle system only had access to the top 1000 translation hypotheses produced by MEMT for each sentence. While this does not guarantee finding the best translation that the decoder can produce, this method provides a good approximation. We also ran the oracle experiment only on the first 140 sentences of the test sets due to time constraints.

All the system performances are measured using the METEOR evaluation metric (Lavie, Sagae et al., 2004). METEOR was chosen since, unlike the more commonly used BLEU metric (Papineni et al., 2002), it provides reasonably reliable scores for individual sentences. This property is essential in order to run our oracle experiments. METEOR produces scores in the range of [0,1], based on a combination of unigram precision, unigram recall and an explicit penalty related to the average length of matched segments between the evaluated translation and its reference.

## 4 Results

| **System** | **METEOR Score** |
|---|---|
| System A | 0.4241 |
| System B | 0.4231 |
| System C | 0.4405 |
| Choosing best original translation | 0.4432 |
| **MEMT System** | **0.5183** |

**Table 1: METEOR Scores on TIDES 2003 Dataset**

On the 2003 TIDES data, the three original systems had similar METEOR scores. Table 1 shows the scores of the three systems, with their names obscured to protect their privacy. Also shown are the score of MEMT's output and the score of the oracle system that chooses the best original translation on a sentence-by-sentence basis. The score of the MEMT system is significantly better than any of the original systems, and the sentence oracle.

On the first 140 sentences, the oracle system that selects the best hypothesis translation generated by the MEMT generator has a METEOR score of 0.5883. This indicates that the scoring algorithm used to select the final MEMT output can be significantly further improved.

## 5 Conclusions and Future Work

Our MEMT algorithm shows consistent improvement in the quality of the translation compared any of the original systems. It scores better than an "oracle" that chooses the best original translation on a sentence-by-sentence basis. Furthermore, our MEMT algorithm produces hypotheses that are of yet even better quality, but our current scoring algorithm is not yet able to effectively select the best hypothesis. The focus of our future work will thus be on identifying features that support improved hypothesis scoring.

## Acknowledgments

This research work was partly supported by a grant from the US Department of Defense. The word alignment matcher was developed by Satanjeev Banerjee. We wish to thank Robert Frederking, Ralf Brown and Jaime Carbonell for their valuable input and suggestions.

## References

Bangalore, S., G.Bordel, and G. Riccardi (2001). Computing Consensus Translation from Multiple Machine Translation Systems. *In Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop* (ASRU-2001), Italy.

Fiscus, J. G.(1997). A Post-processing System to Yield Reduced Error Word Rates: Recognizer Output Voting Error Reduction (ROVER). *In IEEE Workshop on Automatic Speech Recognition and Understanding* (ASRU-1997).

Frederking, R. and S. Nirenburg. Three Heads are Better than One. In Proceedings of the Fourth Conference on Applied Natural Language Processing (ANLP-94), Stuttgart, Germany, 1994.

Hogan, C. and R.E.Frederking (1998). An Evaluation of the Multi-engine MT Architecture. *In Proceedings of the Third Conference of the Association for Machine Translation in the Americas*, pp. 113-123. Springer-Verlag, Berlin .

Lavie, A., K. Probst, E. Peterson, S. Vogel, L.Levin, A. Font-Llitjos and J. Carbonell (2004). A Trainable Transfer-based Machine Translation Approach for Languages with Limited Resources. *In Proceedings of Workshop of the European Association for Machine Translation* (EAMT-2004), Valletta, Malta.

Lavie, A., K. Sagae and S. Jayaraman (2004). The Significance of Recall in Automatic Metrics for MT Evaluation. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas* (AMTA-2004), Washington, DC.

Papineni, K., S. Roukos, T. Ward and W-J Zhu (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics* (ACL-2002), Philadelphia, PA.

Tidhar, Dan and U. Küssner (2000). Learning to Select a Good Translation. *In Proceedings of the 17th conference on Computational linguistics* (COLING 2000), Saarbrücken, Germany.

# SenseClusters: Unsupervised Clustering and Labeling of Similar Contexts

**Anagha Kulkarni** and **Ted Pedersen**
Department of Computer Science
University of Minnesota
Duluth, MN 55812
{kulka020,tpederse}@d.umn.edu
http://senseclusters.sourceforge.net

**Abstract**

SenseClusters is a freely available system that identifies similar contexts in text. It relies on lexical features to build first and second order representations of contexts, which are then clustered using unsupervised methods. It was originally developed to discriminate among contexts centered around a given target word, but can now be applied more generally. It also supports methods that create descriptive and discriminating labels for the discovered clusters.

## 1 Introduction

SenseClusters seeks to group together units of text (referred to as contexts) that are similar to each other using lexical features and unsupervised clustering.

Our initial work (Purandare and Pedersen, 2004) focused on word sense discrimination, which takes as input contexts that each contain a given target word, and produces as output clusters that are presumed to correspond to the different senses of the word. This follows the hypothesis of (Miller and Charles, 1991) that words that occur in similar contexts will have similar meanings.

We have shown that these methods can be extended to proper name discrimination (Pedersen et al., 2005). People, places, or companies often share the same name, and this can cause a considerable amount of confusion when carrying out Web search or other information retrieval applications. Name discrimination seeks to group together the contexts that refer to a unique underlying individual, and allow the user to recognize that the same name is being used to refer to multiple entities.

We have also extended SenseClusters to cluster contexts that are not centered around any target word, which we refer to as *headless clustering*. Automatic email categorization is an example of a headless clustering task, since each message can be considered a context. SenseClusters will group together messages if they are similar in content, without requiring that they share any particular target word between them.

We are also addressing a well known limitation to unsupervised clustering approaches. After clustering contexts, it is often difficult to determine what underlying concepts or entities each cluster represents without manually inspecting their contents. Therefore, we are developing methods that automatically assign *descriptive* and *discriminating* labels to each discovered cluster that provide a characterization of the contents of the clusters that a human can easily understand.

## 2 Clustering Methodology

We begin with the collection of contexts to be clustered, referred to as the test data. These may all include a given target word, or they may be headless contexts. We can select the lexical features from the test data, or from a separate source of data. In either case, the methodology proceeds in exactly the same way.

SenseClusters is based on lexical features, in particular unigrams, bigrams, co–occurrences, and target co–occurrences. Unigrams are single words that occur more than five times, bigrams are ordered pairs of words that may have intervening words between them, while co-occurrences are simply unordered bigrams. Target co-occurrences are those co–occurrences that include the given target word. We select bigrams and co–occurrences that occur more than five times, and that have a log–likelihood ratio of more than 3.841, which signifies a 95% level of certainty that the two words are not independent. We do not allow unigrams to be stop words, and we eliminate any bigram or co–occurrence feature that includes one or more stop words.

Previous work in word sense discrimination has shown that contexts of an ambiguous word can be effectively represented using first order (Pedersen and Bruce, 1997) or second order (Schütze, 1998) representations. SenseClusters provides extensive support for both, and allows for them to be applied in a wider range of problems.

In the first order case, we create a context (rows) by lexical features (columns) matrix, where the features may be any of the above mentioned types. The cell values in this matrix record the frequencies of each feature occurring in the context represented by a given row. Since most lexical features only occur a small number of times (if at all) in each context, the resulting matrix tends to be very sparse and nearly binary. Each row in this matrix forms a vector that represents a context. We can (optionally) use Singular Value Decomposition (SVD) to reduce the dimensionality of this matrix. SVD has the effect of compressing a sparse matrix by combining redundant columns and eliminating noisy ones. This allows the rows to be represented with a smaller number of hopefully more informative columns.

In the second order context representation we start with creating a word by word co-occurrence matrix where each row represent the first word and the columns represent the second word of either bigram or co–occurrence features previously identified. If the features are bigrams then the word matrix is asymmetric whereas for co-occurrences it is symmetric and the rows and columns do not suggest any ordering. In either case, the cell values indicate how often the two words occur together, or contains their log–likelihood score of associativity. This matrix is large and sparse, since most words do not co–occur with each other. We may optionally apply SVD to this co-occurrence matrix to reduce its dimensionality. Each row of this matrix is a vector that represents the given word at the row via its co–occurrence characteristics. We create a second order representation of a context by replacing each word in that context with its associated vector, and then averaging together all these word vectors. This results in a single vector that represents the overall context.

For contexts with target words we can restrict the number of words around the target word that are averaged for the creation of the context vector. In our name discrimination experiments we limit this scope to five words on either side of the target word which is based on the theory that words nearer to the target word are more related to it than the ones that are farther away.

The goal of the second order context representation is to capture indirect relationships between words. For example, if the word *Dictionary* occurs with *Words* but not with *Meanings*, and *Words* occurs with *Meanings*, then the words *Dictionary* and *Meanings* are second order co-occurrences via the first order co-occurrence of *Words*.

In either the first or second order case, once we have each context represented as a vector we proceed with clustering. We employ the hybrid clustering method known as Repeated Bisections, which offers nearly the quality of agglomerative clustering at the speed of partitional clustering.

## 3 Labeling Methodology

For each discovered cluster, we create a *descriptive* and a *discriminating* label, each of which is made up of some number of bigram features. These are identified by treating the contexts in each cluster as a separate corpora, and applying our bigram feature selection methods as described previously on each of them.

*Descriptive* labels are the top N bigrams according to the log–likelihood ratio. Our goal is that these labels will provide clues as to the general nature of the contents of a cluster. The *discriminating* labels are any *descriptive* labels for a cluster that are not *descriptive* labels of another cluster. Thus, the *discriminating* label may capture the content that separates one cluster from another and provide a more

Table 1: Name Discrimination (F-measure)

| 2-Way Name(M);+ | MAJ. (N) | O1 k=2 | O2 k=2 |
|---|---|---|---|
| AAIRLINES(1075); TCRUISE(1075) | 50.0 (2150) | **66.6** | 58.8 |
| AAIRLINES(3966); HPACKARD(3690) | 51.7 (7656) | **61.7** | 59.6 |
| BGATES(1981); TCRUISE(1075) | 64.8 (3056) | 63.4 | 53.8 |
| BSPEARS(1380); GBUSH(1380) | 50.0 (2760) | 56.6 | **65.8** |
| 3-Way Name (M);+ | | k=3 | k=3 |
| AAIRLINES(2500); HPACKARD(2500); BMW(2500); | 33.3 (7500) | 41.4 | **45.1** |
| AAIRLINES(1300); HPACKARD(1300); BSPEARS(1300); | 33.3 (3900) | **46.0** | **45.3** |
| BGATES(1075); TCRUISE(1075); GBUSH(1075) | 33.3 (3225) | **53.7** | **53.6** |

Table 2: Email Categorization (F-measure)

| Newsgroup(M);+ | MAJ. (N) | O1 k=2 | O2 k=2 |
|---|---|---|---|
| comp.graphics(389); misc.forsale(390) | 50.1 (779) | **61.1** | **63.9** |
| comp.graphics(389); talk.pol.mideast(376) | 50.8 (756) | **73.6** | 54.8 |
| rec.motorcycles(398); sci.crypt(396) | 50.13 (794) | **83.1** | 60.5 |
| rec.sport.hockey(399); soc.relig.christian(398) | 50.1 (797) | **77.6** | 58.5 |
| sci.electronics(393); soc.relig.christian(398) | 50.3 (791) | **67.8** | 52.3 |

detailed level of information.

## 4 Experimental Data

We evaluate these methods on proper name discrimination and email (newsgroup) categorization.

For name discrimination we use the 700 million word New York Times portion of the English GigaWord corpus as the source of contexts. While there are many ambiguous names in this data, it is difficult to evaluate the results of our approach given the absence of a disambiguated version of the text. Thus, we automatically create ambiguous names by conflating the occurrences associated with two or three relatively unambiguous names into a single obfuscated name.

For example, we combine *Britney Spears* and *George Bush* into an ambiguous name *Britney Bush*, and then see how well SenseClusters is able to create clusters that reflect the true underlying identity of the conflated name.

Our email experiments are based on the 20-NewsGroup Corpus of USENET articles. This is a collection of approximately 20,000 articles that have been taken from 20 different newsgroups. As such they are already classified, but since our methods are unsupervised we ignore this information until it is time to evaluate our approach. We present results that make two way distinctions between selected pairs of newsgroups.

## 5 Experimental Results and Discussion

Table 1 presents the experimental results for 2-way and 3-way name discrimination experiments, and Table 2 presents results for a 2-way email categorization experiment. The results are reported in terms of the F-measure, which is the harmonic mean of precision and recall.

The first column in both tables indicates the possible names or newgroups, and the number of contexts associated with each. The next column indicates the percentage of the majority class (MAJ.) and count (N) of the total number of contexts for the names or newsgroups. The majority percentage provides a simple baseline for level of performance, as this is the F–measure that would be achieved if every context were simply placed in a single cluster. We refer to this as the unsupervised majority classifier.

The next two columns show the F–measure associated with the order 1 and order 2 representations of context, with all other options being held constant. These experiments used bigram features, SVD was performed as appropriate for each representation, and the method of Repeated Bisections was used for clustering.

Table 3: Cluster Labels (for Table 1)

| True Name | Created Labels |
|---|---|
| CLUSTER 0: AMERICAN AIRLINES | **Flight 11**, **Flight 587**, **Sept 11**, **Trade Center**, **World Trade**, Los Angeles, New York |
| CLUSTER 1: TOM CRUISE | **Jerry Maguire**, **Mission Impossible**, **Minority Report**, **Tom Cruise**, **Penelope Cruz**, **Nicole Kidman**, **United Airlines**, **Vanilla Sky**, Los Angeles, New York |
| CLUSTER 0: GEORGE BUSH | **George Bush** , **George W**, **Persian Gulf**, **President**, **U S**, **W Bush**, **former President**, **lifting feeling**, White House |
| CLUSTER 1: BILL GATES | **Chairman** , **Microsoft** , **Microsoft Chairman**, **co founder**, **News Service**, **operating system**, **chief executive**, White House |
| CLUSTER 2: TOM CRUISE | **Jerry Maguire**, **Mission Impossible**, **Minority Report**, **Al Gore**, **New York** , **Nicole Kidman**, **Penelope Cruz**, **Vanilla Sky**, **Ronald Reagan**, White House |

Finally, note that the number of clusters to be discovered must be provided by the user. In these experiments we have taken the best case approach and asked for a number of clusters equal to that which actually exists. We are currently working to develop methods that will automatically stop at an optimal number of clusters, to avoid setting this value manually.

In general all of our results significantly improve upon the majority classifier, which suggests that the clustering of contexts is successfully discriminating among ambiguous names and uncategorized email.

Table 3 shows the *descriptive* and *discriminating* labels assigned to the 2–way experimental case of *American Airlines* and *Tom Cruise*, as well as the 3–way case of *George Bush*, *Bill Gates* and *Tom Cruise*. The bold face labels are those that serve as both *descriptive* and *discriminating* labels. The fact that most labels serve both roles suggests that the highest ranked bigrams in each cluster were also unique to that cluster. The normal font indicates labels that are only *descriptive*, and are shared between multiple clusters. There are only a few such cases, for example *White House* happens to be a significant bigram in all three of the clusters in the 3–way case. There were no labels that were exclusively *discriminating* in these experiments, suggesting that the clusters are fairly clearly distinguished.

Please note that some labels include unigrams (e.g., *President* for *George Bush*). These are created from bigrams where the other word is the conflated form, which is not included in the labels since it is by definition ambiguous.

## 6 Acknowledgements

This research is partially supported by a National Science Foundation Faculty Early CAREER Development Award (#0092784).

## References

G.A. Miller and W.G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

T. Pedersen and R. Bruce. 1997. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, Providence, RI, August.

T. Pedersen, A. Purandare, and A. Kulkarni. 2005. Name discrimination by clustering similar contexts. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 220–231, Mexico City, February.

A. Purandare and T. Pedersen. 2004. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 41–48, Boston, MA.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

# A Flexible Stand-Off Data Model with Query Language for Multi-Level Annotation

**Christoph Müller**
EML Research gGmbH
Villa Bosch
Schloß-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
mueller@eml-research.de

## Abstract

We present an implemented XML data model and a new, simplified query language for multi-level annotated corpora. The new query language involves automatic conversion of queries into the underlying, more complicated MMAXQL query language. It supports queries for sequential and hierarchical, but also associative (e.g. coreferential) relations. The simplified query language has been designed with non-expert users in mind.

## 1 Introduction

Growing interest in richly annotated corpora is a driving force for the development of annotation tools that can handle multiple levels of annotation. We find it crucial in order to make full use of the potential of multi-level annotation that individual annotation levels be treated as *self-contained modules* which are independent of other annotation levels. This independence should also include the storing of each level in a separate file. If these principles are observed, annotation data management (incl. level addition, removal and replacement, but also conversion into and from other formats) is greatly facilitated.

The way to keep individual annotation levels independent of each other is by defining each with direct reference to the underlying basedata, i.e. the text or transcribed speech. Both sequential and hierarchical (i.e. embedding or dominance) relations between markables on different levels are thus only expressed *implicitly*, viz. by means of the relations of their basedata elements.

While it has become common practice to use the stand-off mechanism to relate several annotation levels to one basedata file, it is also not uncommon to find this mechanism applied for relating markables to other markables (on a different or the same level) directly, expressing the relation between them *explicitly*. We argue that this is unfavourable not only with respect to annotation data management (cf. above), but also with respect to *querying*: Users should not be required to formulate queries in terms of structural properties of data representation that are irrelevant for their query. Instead, users should be allowed to relate markables from all levels in a fairly unrestricted and ad-hoc way. Since querying is thus considerably simplified, exploratory data analysis of annotated corpora is facilitated for all users, including non-experts.

Our multi-level annotation tool MMAX2[1] (Müller & Strube, 2003) uses implicit relations only. Its query language MMAXQL is rather complicated and not suitable for naive users. We present an alternative query method consisting of a simpler and more intuitive query language and a method to generate MMAXQL queries from the former. The new, *simplified* MMAXQL can express a wide range of queries in a concise way, including queries for associative relations representing e.g. coreference.

## 2 The Data Model

We propose a stand-off data model implemented in XML. The basedata is stored in a simple XML file

[1]The current release version of MMAX2 can be downloaded at http://mmax.eml-research.de.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE words SYSTEM "words.dtd">
<words>
 ...
<word id="word_1064">My</word>
<word id="word_1065">,</word>
<word id="word_1066">uh</word>
<word id="word_1067">,</word>
<word id="word_1068">cousin</word>
<word id="word_1069">is</word>
<word id="word_1070">a</word>
<word id="word_1071">F</word>
<word id="word_1072">B</word>
<word id="word_1073">I</word>
<word id="word_1074">agent</word>
<word id="word_1075">down</word>
<word id="word_1076">in</word>
<word id="word_1077">Miami</word>
<word id="word_1078">.</word>
 ...
<word id="word_1085">she</word>
...
</words>
```

Figure 1: `basedata` file (extract)

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE markables SYSTEM "markables.dtd">
<markables xmlns="www.eml.org/NameSpaces/utterances">
 ...
<markable id="markable_116" span="word_1064..word_1078"/>
 ...
</markables>
```

Figure 2: `utterances` level file (extract)

which serves to identify individual tokens[2] and associate an ID with each (Figure 1).

In addition, there is one XML file for each annotation level. Each level has a unique, descriptive name, e.g. `utterances` or `pos`, and contains annotations in the form of <`markable`> elements. In the most simple case, a markable only identifies a sequence (i.e. *span*) of basedata elements (Figure 2).

Normally, however, a markable is also associated with arbitrarily many user-defined attribute-value pairs (Figure 3, Figure 4). Markables can also be discontinuous, like `markable_954` in Figure 4.

For each level, admissible attributes and their values are defined in a separate annotation scheme file (not shown, cf. Müller & Strube (2003)). Freetext attributes can have any string value, while nominal attributes can have one of a (user-defined) closed set of possible values. The data model also supports associative relations between markables: *Markable set* relations associate arbitrarily many markables with each other in a transitive, undirected way. The `coref_class` attribute in Figure 4 is an example of how such a relation can be used to represent a coreferential relation between markables (here: `markable_954` and `markable_963`, rest of set

[2]Usually words, but smaller elements like morphological units or even characters are also possible.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE markables SYSTEM "markables.dtd">
<markables xmlns="www.eml.org/NameSpaces/pos">
 ...
<markable id="markable_665" span="word_1064" pos="PRP$"/>
<markable id="markable_666" span="word_1065" pos=","/>
<markable id="markable_667" span="word_1066" pos="UH"/>
<markable id="markable_668" span="word_1067" pos=","/>
<markable id="markable_669" span="word_1068" pos="NN"/>
<markable id="markable_670" span="word_1069" pos="VBZ"/>
<markable id="markable_671" span="word_1070" pos="DT"/>
<markable id="markable_672" span="word_1071" pos="NNP"/>
<markable id="markable_673" span="word_1072" pos="NNP"/>
<markable id="markable_674" span="word_1073" pos="NNP"/>
<markable id="markable_675" span="word_1074" pos="NN"/>
<markable id="markable_676" span="word_1075" pos="IN"/>
<markable id="markable_677" span="word_1076" pos="IN"/>
<markable id="markable_678" span="word_1077" pos="NNP"/>
<markable id="markable_679" span="word_1078" pos="."/>
 ...
<markable id="markable_686" span="word_1085" pos="PRP"/>
...
</markables>
```

Figure 3: `pos` level file (extract)

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE markables SYSTEM "markables.dtd">
<markables xmlns="www.eml.org/NameSpaces/ref_exp">
 ...
<markable id="markable_953" span="word_1064" type="poss_det"/>
<markable id="markable_954" span="word_1064,word_1068" type="np"
  coref_class="set_3"/>
<markable id="markable_955" span="word_1070..word_1074" type="np"/>
<markable id="markable_956" span="word_1071..word_1073" type="pn"/>
<markable id="markable_957" span="word_1077" type="pn"/>
 ...
<markable id="markable_963" span="word_1085" type="pron"
  coref_class="set_3"/>
...
</markables>
```

Figure 4: `ref_exp` level file (extract)

not shown). *Markable pointer* relations associate with one markable (the *source*) one or more *target* markables in an intransitive, directed fashion.

## 3 Simplified MMAXQL

Simplified MMAXQL is a variant of the MMAXQL query language. It offers a simpler and more concise way to formulate certain types of queries for multi-level annotated corpora. Queries are automatically converted into the underlying query language and then executed. A query in simplified MMAXQL consists of a sequence of *query tokens* which are combined by means of *relation operators*. Each query token queries exactly one basedata element (i.e. word) or one markable.

### 3.1 Query Tokens

**Basedata** elements can be queried by matching regular expressions. Each basedata query token consists of a regular expression in single quotes, which must *exactly* match one basedata element. The query

```
'[Tt]he'
```

matches all definite articles, but not e.g. *ether* or *there*. For the latter two words to also match, wildcards have to be used:

```
'.*[Tt]he.*'
```

Sequences of basedata elements can be queried by simply concatenating several space-separated[3] tokens. The query

```
'[Tt]he  [A-Z].+'
```

will match sequences consisting of a definite article and a word beginning with a capital letter.

**Markables** are the carriers of the actual annotation information. They can be queried by means of string matching and by means of attribute-value combinations. A markable query token has the form

```
string/conditions
```

where `string` is an optional regular expression and `conditions` specifies which attribute(s) the markable should match. The most simple 'condition' is just the name of a markable level, which will match all markables on that level. If a regular expression is also supplied, the query will return only the matching markables. The query

`[Aa]n?\s.*/ref_exp`[4]

will return all markables from the `ref_exp` level beginning with the indefinite article.

The `conditions` part of a markable query token can indeed be much more complex. A main feature of simplified MMAXQL is that redundant parts of conditions can **optionally** be left out, making queries very concise. For example, the *markable level name* can be left out if the name of the *attribute* accessed by the query is unique across all active markable levels. Thus, the query

```
/!coref_class=empty
```

can be used to query markables from the `ref_exp` level which have a non-empty value in the `coref_class` attribute, granted that only one attribute of this name exists.[5] The same applies to the names of *nominal attributes* if the *value* specified in the query unambiguously points to this attribute. Thus, the query

```
/pn
```

can be used to query markables from the `pos` level which have the value `pn`, granted that there is exactly one nominal attribute with the possible value `pn`. Several conditions can be combined into one query token. Thus, the query

```
/{poss_det,pron},!coref_class=empty
```

returns all markables from the `ref_exp` level that are either possessive determiners or pronouns and that are part in some coreference set.[6]

## 3.2 Relation Operators

The whole point of querying corpora with multi-level annotation is to relate markables from different levels to each other. The reference system with respect to which the relation between different markables is established is the sequence of basedata elements, which is the same for all markables on all levels. Since this bears some resemblance to different *events* occurring in several *temporal* relations to each other, we (like also Heid et al. (2004), among others) adopt this as a metaphor for expressing the **sequential** and **hierarchical** relations between markables, and we use a set of relation operators that is inspired by (Allen, 1991). This set includes (among others) the operators `before`, `meets` (default), `starts`, `during/in`, `contains/dom`, `equals`, `ends`, and some inverse relations. The following examples give an idea of how individual query tokens can be combined by means of relation operators to form complex queries. The example uses the ICSI meeting corpus of spoken multi-party dialogue.[7] This corpus contains, among others, a `segment` level with markables roughly corresponding to speaker turns, and a `meta` level containing markables representing e.g. pauses, emphases, or sounds like breathing or mike noise. These two levels and the basedata level can be combined to retrieve instances of *you know* that occur in segments spoken by female speakers[8] which also contain a pause or an emphasis:

```
'[Yy]ou know' in (/participant={f.*} dom /{pause,emphasis})
```

[3]Using the fact that `meets` is the default relation operator, cf. Section 3.2.

[4]The space character in the regular expression must be masked as `\s` because otherwise it will be interpreted as a query token separator.

[5]If this condition does not hold, attribute names can be disambiguated by prepending the markable level name.

[6]The curly braces notation is used to specify several OR-connected values for a single attribute, while a comma *outside* curly braces is used to AND-connect several conditions relating to different attributes.

[7]Obtained from the LDC and converted into MMAX2 format, preserving all original information.

[8]The first letter of the `participant` value encodes the speaker's gender.

Relation operators for **associative** relations (i.e. markable set and markable pointer) are `nextpeer`, `anypeer` and `nexttarget`, `anytarget`, respectively. Assuming the sample data from Section 2, the query

```
/ref_exp nextpeer:coref_class /ref_exp
```

retrieves pairs of anaphors (right) and their direct antecedents (left). The query can be modified to

```
/ref_exp nextpeer:coref_class (/ref_exp equals /pron)
```

to retrieve only anaphoric *pronouns* and their direct antecedents.

If a query is too complex to be expressed as a single query token sequence, *variables* can be used to store intermediate results of sub-queries. The following query retrieves pairs of utterances (incl. the referring expressions embedded into them) that are more than 30 tokens[9] apart, and assigns the resulting 4-tuples to the variable `$distant_utts`.

```
(/utterances dom /ref_exp) before:31- (/utterances dom /ref_exp)
   -> $distant_utts
```

The next query accesses the second and last column in the temporary result (by means of the zero-based column index) and retrieves those pairs of anaphors and their direct antecedents that occur in utterances that are more than 30 tokens apart:

```
$distant_utts.1 nextpeer:coref_class $distant_utts.3
```

## 4 Related Work

In the EMU speech database system (Cassidy & Harrington, 2001) the hierarchical relation between levels has to be made explicit. Sequential and hierarchical relations can be queried like with simplified MMAXQL, with the difference that e.g. for sequential queries, the elements involved must come from the same level. Also, the result of a hierarchical query always only contains either the parent or child element. The EMU data model supports an association relation (similar to our markable pointer) which can be queried using a => operator.

Annotation Graphs (Bird & Liberman, 2001) identify elements on various levels as arcs connecting two points on a time scale shared by all levels. Relations between elements are thus also represented implicitly. The model can also express a binary association relation. The associated Annotation Graph query language (Bird et al., 2000) is very explicit, which makes it powerful but at the same time possibly too demanding for naive users.

[9]A means to express distance in terms of markables is not yet available, cf. Section 5.

The NITE XML toolkit (Carletta et al., 2003) defines a data model that is close to our model, although it allows to express hierarchical relations explicitly. The model supports a labelled pointer relation which can express one-to-many associations. The associated query language NXT Search (Heid et al., 2004) is a powerful declarative language for querying diverse relations (incl. pointers), supporting quantification and constructs like `forall` and `exists`.

## 5 Future Work

We work on support for queries like 'pairs of referring expressions that are a certain number of referring expressions apart'. We also want to include wild cards and proximity searches, and support for automatic markable creation from query results.

## Acknowledgements

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany.

## References

Allen, James (1991). Time and time again. *International Journal of Intelligent Systems*, 6(4):341–355.

Bird, Steven, Peter Buneman & Wang-Chiew Tan (2000). Towards a query language for annotation graphs. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece, 31 May-June 2, 2000, pp. 807–814.

Bird, Steven & Mark Liberman (2001). A formal framework for linguistic annotation. *Speech Communication*, 33:23–60.

Carletta, Jean, Stefan Evert, Ulrich Heid, Jonathan Kilgour, J. Robertson & Holger Voormann (2003). The NITE XML toolkit: flexible annotation for multi-modal language data. *Behavior Research Methods, Instruments, and Computers*, 35:353–363.

Cassidy, Steve & Jonathan Harrington (2001). Multi-level annotation in the EMU speech database management system. *Speech Communication*, 33:61–78.

Heid, Ulrich, Holger Voormann, Jan-Torsten Milde, Ulrike Gut, Katrin Erk & Sebastian Pado (2004). Querying both time-aligned and hierarchical corpora with NXT search. In *Proceedings of the 4th International Conference on Language Resources and Evaluation,* Lisbon, Portugal, 26-28 May, 2004, pp. 1455–1458.

Müller, Christoph & Michael Strube (2003). Multi-level annotation in MMAX. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue,* Sapporo, Japan, 4-5 July 2003, pp. 198–207.

# HAHAcronym: A Computational Humor System

**Oliviero Stock** and **Carlo Strapparava**
ITC-irst, Istituto per la Ricerca Scientifica e Tecnologica
I-38050 Trento, ITALY
{stock, strappa}@itc.it

## Abstract

Computational humor will be needed in interfaces, no less than other cognitive capabilities. There are many practical settings where computational humor will add value. Among them there are: business world applications (such as advertisement, e-commerce, etc.), general computer-mediated communication and human-computer interaction, increase in the friendliness of natural language interfaces, educational and edutainment systems. In particular in the educational field it is an important resource for getting selective attention, help in memorizing names and situations etc. And we all know how well it works with children.

Automated humor production in general is a very difficult task but we wanted to prove that some results can be achieved even in short time. We have worked at a concrete limited problem, as the core of the European Project HAHAcronym. The main goal of HAHAcronym has been the realization of an acronym ironic reanalyzer and generator as a proof of concept in a focalized but non restricted context. To implement this system some general tools have been adapted, or developed for the humorous context. Systems output has been submitted to evaluation by human subjects, with a very positive result.

## 1 Introduction

Society needs humor, not just for entertainment. In the current business world, humor is considered to be so important that companies may hire humor consultants. Humor can be used "to criticize without alienating, to defuse tension or anxiety, to introduce new ideas, to bond teams, ease relationships and elicit cooperation".

As far as human-computer interfaces are concerned, in the future we will demand naturalness and effectiveness that require the incorporation of models of possibly all human cognitive capabilities, including the handling of humor (Stock, 1996). There are many practical settings where computational humor will add value. Among them there are: business world applications (such as advertisement, e-commerce, etc.), general computer-mediated communication and human-computer interaction, increase in the friendliness of natural language interfaces, educational and edutainment systems.

Not necessarily applications need to emphasize interactivity. For instance there are important prospects for humor in automatic information presentation. In the Web age presentations will become more and more flexible and personalized and will require humor contributions for electronic commerce developments (e.g. product promotion, getting selective attention, help in memorizing names etc) more or less as it happened in the world of advertisement within the old broadcast communication.

Little published research exists on whether humor is valuable in task-oriented human-computer interaction (HCI). However (Morkes et al., 1999) did some experiments concerning the effects of humor in HCI and computer-mediated communication situations. Especially in computer-mediated communication tasks, participants who received jokes rated the "person" or computer they worked with as more likable and competent, reported greater cooperation, joked back more often etc. The experiments show that, humor enhances the likeability of an interface "without distracting users".

There has been a considerable amount of research on linguistics of humor and on theories of semantics or pragmatics of humor (Attardo, 1994). Within the artificial intelligence community, most writing on humor has been speculative (Hofstadter et al., 1989). Minsky (Minsky, 1980) made some preliminary remarks about formalizing some kind of humor within an artificial intelligence/cognitive science perspective. He refined Freud's notion that humor is a way of bypassing our mental "censors" which control inappropriate thoughts and feelings (Freud, 1905). So far, very limited effort has been put on building computational humor prototypes. The few existing ones are concerned with rather simple tasks, normally in limited domains. Probably the most important attempt to create a computational humor prototype is the work of Binsted and Ritchie (Binsted and Ritchie, 1994). They have devised a model of the semantic and syntactic regularities underlying some of the simplest types of punning riddles. A punning riddle is a question-answer riddle that uses phonological ambiguity. The three main strategies used to create phonological ambiguity are syllable substitution, word substitution and metathesis. In general, the constructive approaches are mostly inspired by the incongruity theory (Raskin, 1985), interpreted at various level of refinement. The incongruity theory focuses on the element of surprise. It states that humor is created out of a conflict between what is expected and what actually occurs when the humorous utterance or story is completed. In verbal humor this means that at some level, different interpretations of material must be possible (and some not detected before the culmination of the humorous process) or various pieces of material must cause perception of specific forms of opposition. Natural language processing research has often dealt with ambiguity in language. A common view is that ambiguity is an obstacle for deep comprehension. Exactly the opposite is true here.

The work presented here refers to HAHAcronym, the first European project devoted to computational humor (EU project IST-2000-30039), part of the Future Emerging Technologies section of the Fifth European Framework Program. The main goal of HAHAcronym was the realization of an acronym ironic re-analyzer and generator as a proof of concept in a focalized but non restricted context. In the first case the system makes fun of existing acronyms, in the second case, starting from concepts provided by the user, it produces new acronyms, constrained to be words of the given language. And, of course, they have to be funny.

HAHAcronym, fully described in (Stock and Strapparava, 2003) (Stock and Strapparava, 2005), is based on various resources for natural language processing, adapted for humor. Many components are present but simplified with respect to more complex scenarios and some general tools have been developed for the humorous context. A fundamental tool is an incongruity detector/generator: in practice there is a need to detect semantic mismatches between expected sentence meaning and other readings, along some specific dimension (i.e. in our case the acronym and its context).

## 2 The HAHAcronym project

The realization of an acronym re-analyzer and generator was proposed to the European Commission as a project that we would be able to develop in a short period of time (less than a year), that would be meaningful, well demonstrable, that could be evaluated along some pre-decided criteria, and that was conducive to a subsequent development in a direction of potential applicative interest. So for us it was essential that:

1. the work could have many components of a larger system, simplified for the current setting;
2. we could reuse and adapt existing relevant linguistic resources;
3. some simple strategies for humor effects could be experimented.

One of the purposes of the project was to show that using "standard" resources (with some extensions and modifications) and suitable linguistic theories of humor (i.e. developing specific algorithms that implement or elaborate theories), it is possible to implement a working prototype. For that, we have taken advantage of specialized thesauri and repositories and in particular of WORDNET DOMAINS, an extension developed at ITC-irst of the well-known English WORDNET. In WORDNET DOMAINS, synsets are annotated with subject field codes (or domain labels), e.g. MEDICINE, ARCHITECTURE, LITERATURE,... In particular for HAHAcronym, we have modelled an independent structure of domain opposition, such as RELIGION vs. TECHNOLOGY, SEX vs. RELIGION, etc..., as a basic resource for the incongruity generator.
Other important computational tools we have used are: a parser for analyzing input syntactically and a syntactic generator of acronyms; general lexical resources, e.g. acronym grammars, morphological analyzers, rhyming dictionaries, proper nouns databases, a dictionary of hyperbolic adjectives/adverbs.

### 2.1 Implementation

To get an ironic or profaning re-analysis of a given acronym, the system follows various steps and relies on a number of strategies. The main elements of the algorithm can be schematized as follows:

- acronym parsing and construction of a logical form
- choice of what to keep unchanged (for example the head of the highest ranking NP) and what to modify (for example the adjectives)
- look for possible, initial letter preserving, substitutions
  - using semantic field oppositions;
  - reproducing rhyme and rhythm (the modified acronym should sound as similar as possible to the original one);
  - for adjectives, reasoning based mainly on antonym clustering and other semantic relations in WORDNET.

Making fun of existing acronyms amounts to basically using irony on them, desecrating them with some unexpectedly contrasting but otherwise consistently sounding expansion.

As far as acronym generation is concerned, the problem is more complex. We constrain resulting acronyms to be words of the dictionary. The system takes in input some concepts (actually synsets, so that input to this system can result from some other processing, for instance sentence interpretation) and some minimal structural indication, such as the semantic head. The primary strategy of the system is to consider as potential acronyms words that are in ironic relation with input concepts. Structures for the acronym expansion result from the specified head indication and the grammar. Semantic reasoning and navigation over WORDNET, choice of specific word realizations, including morphosyntactic variations, constrain the result. In this specific strategy, ironic reasoning is developed mainly at the level of acronym choice and in the incongruity resulting in relation to the coherently combined words of the acronym expansion.

## 3 Examples and Evaluation

Here below some examples of acronym re-analysis are reported. As far as semantic field opposition is concerned, we have slightly biased the system towards the domains FOOD, RELIGION, and SEX. For each example we report the original acronym and the re-analysis.

`ACM` - Association for Computing Machinery

→ `Association for Confusing Machinery`

`FBI` - Federal Bureau of Investigation

→ `Fantastic Bureau of Intimidation`

`PDA` - Personal Digital Assistant

→ `Penitential Demoniacal Assistant`

`IJCAI` - International Joint Conference on Artificial Intelligence

→ `Irrational Joint Conference on Antenuptial Intemperance`

→ `Irrational Judgment Conference on Artificial Indolence`

`ITS` - Intelligent Tutoring Systems

→ `Impertinent Tutoring Systems`

→ `Indecent Toying Systems`

As far as generation from scratch is concerned, a main concept and some attributes (in terms of synsets) are given as input to the system. Here below we report some examples of acronym generation.

Main concept: *tutoring*; Attribute: *intelligent*

**`FAINT`** - Folksy Acritical Instruction for Nescience Teaching

**`NAIVE`** - Negligent At-large Instruction for Vulnerable Extracurricular-activity

Main concept: *writing*; Attribute: *creative*

**`CAUSTIC`** - Creative Activity for Unconvincingly Sporadically Talkative Individualistic Commercials

We note that the system tries to keep all the expansions of the acronym coherent in the same semantic field of the main concepts. At the same time, whenever possible, it exploits some incongruity in the lexical choices.

Testing the humorous quality of texts or other verbal expressions is not an easy task. There are some relevant studies though, such as (Ruch, 1996). For HAHAcronym an evaluation was set with a group of 30 American university students. They had to evaluate the system production (80 reanalyzed and 80 generated acronyms), along a scale of five levels of amusement (from *very-funny* to *not-funny*). The results were very encouraging. The system performance with humorous strategies and the one without such strategies (i.e. random lexical choices, maintaining only syntactic correctness) were totally different. None of the humorous re-analyses proposed to the students were rejected as completely non-humorous. Almost 70% were rated funny enough (without humorous strategies the figure was less than 8%). In the case of generation of new acronyms results were positive in 53% of the cases.

A curiosity that may be worth mentioning: HAHAcronym participated to a contest about (human) production of best acronyms, organized by RAI, the Italian National Broadcasting Service. The system won a jury's special prize.

## 4 Conclusion

The results of the HAHAcronym project have been positive and a neat prototype resulted, aimed at a very specific task, but operating without restrictions of domain. It turns out that it can be even useful *per se*, but we think that the project opens the way to developments for creative language. We believe that an environment for proposing solutions to advertising professionals can be a realistic practical development of computational humor. In the log run, electronic commerce, for instance, could include flexible and individual-oriented humorous promotion.

## References

S. Attardo. 1994. *Linguistic Theory of Humor*. Mouton de Gruyter, Berlin.

K. Binsted and G. Ritchie. 1994. An implemented model of punning riddles. In *Proc. of the 12$^{th}$ National Conference on Artificial Intelligence (AAAI-94)*, Seattle.

S. Freud. 1905. *Der Witz und Seine Beziehung zum Unbewussten*. Deutike, Leipzig and Vienna.

D. Hofstadter, L. Gabora, V. Raskin, and S. Attardo. 1989. Synopsis of the workshop on humor and cognition. *Humor*, 2(4):293–347.

M. Minsky. 1980. Jokes and the logic of the cognitive unconscious. Technical report, MIT Artificial Intelligence Laboratory. AI memo 603.

J. Morkes, H. Kernal, and C. Nass. 1999. Effects of humor in task-oriented human-computer interaction and computer-mediated communication. *Human-Computer Interaction*, 14:395–435.

V. Raskin. 1985. *Semantic Mechanisms of Humor*. Dordrecht/Boston/Lancaster.

W. Ruch. 1996. Special issue: Measurement approaches to the sense of humor. *Humor*, 9(3/4).

O. Stock and C. Strapparava. 2003. Getting serious about the development of computational humor. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI03)*, Acapulco, Mexico.

O. Stock and C. Strapparava. 2005. The act of creating humorous acronyms. *Applied Artificial Intelligence*, 19(2):137–151, February.

O. Stock. 1996. Password Swordfish: Verbal humor in the interface. In J. Hulstijn and A. Nijholt, editors, *Proc. of International Workshop on Computational Humour (TWLT 12)*, University of Twente, Enschede, Netherlands.

# Organizing English Reading Materials for Vocabulary Learning

**Masao Utiyama, Midori Tanimura** and **Hitoshi Isahara**
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Souraku-gun, Kyoto 619-0289 Japan
{mutiyama,mtanimura,isahara}@nict.go.jp

## Abstract

We propose a method of organizing reading materials for vocabulary learning. It enables us to select a concise set of reading texts (from a target corpus) that contains all the target vocabulary to be learned. We used a specialized vocabulary for an English certification test as the target vocabulary and used English Wikipedia, a free-content encyclopedia, as the target corpus. The organized reading materials would enable learners not only to study the target vocabulary efficiently but also to gain a variety of knowledge through reading. The reading materials are available on our web site.

## 1 Introduction

EFL (English as a foreign language) learners and teachers can easily access a wide range of English reading materials on the Internet. For example, current news stories can be read on web sites such as those for CNN,[1] TIME,[2] or the BBC.[3] Specialized reading materials for EFL learners are also provided on web sites like EFL Reading.[4]

This situation, however, does not mean that EFL learners and teachers can easily select proper texts suited to their specific purposes, for example, learning vocabulary through reading. On the contrary, EFL teachers have to carefully select texts, if they want their students to learn a specialized vocabulary through reading in a particular discipline such as medicine, engineering, or economics. However, it is problematic for teachers to select materials for learning a target vocabulary with short authentic texts.

It is possible to automate this selection process given the target vocabulary to be learned and the target corpus from which texts are gathered (Utiyama et al., 2004). In this research (Utiyama et al., 2004), we used a specialized vocabulary for an English certification test as the target vocabulary and used newspaper articles from *The Daily Yomiuri* as the target corpus. We then organized a set of reading materials, which we called *courseware*[5], using the algorithm in Section 2. The courseware consisted of 116 articles and contained all the target vocabulary. We used the courseware in university English classes from May 2004 to January 2005. We found that the courseware was effective in learning vocabulary (Tanimura and Utiyama, in preparation).

Based on the promising results, our next goal is to distribute courseware (produced with our algorithm) to EFL teachers and learners so that we can receive wider feedback. To this end, the courseware we constructed (Utiyama et al., 2004) is inadequate because it was prepared from The Daily Yomiuri, which is copyrighted. We therefore replaced The Daily Yomiuri with English Wikipedia,[6] a free-content encyclopedia, and developed new course-

[1]http://www.cnn.com/
[2]http://www.time.com/time/
[3]http://www.bbc.co.uk/
[4]http://www.gradedreading.pwp.blueyonder.co.uk/

[5]Courseware usually includes software in addition to other materials. However, in this paper, the term *courseware* is used to refer to the reading materials only.
[6]http://en.wikipedia.org/wiki/Main_Page

ware. It is available on our web site.[7]

In the following, will we first summarize our algorithm and then describe details on the courseware we constructed from English Wikipedia.

## 2 Algorithm

We want to prepare *efficient* courseware for learning a target vocabulary. We defined *efficiency* in terms of the amount of reading materials that must be read to learn a required vocabulary. That is, efficient courseware is as short as possible, while containing the required vocabulary. We used a greedy method to develop the efficient courseware (Utiyama et al., 2004).

Let $C$ be the courseware under development and $V$ be the target vocabulary to be learned. We iteratively select a document (from the target corpus) that has the largest number of new types[8] (types contained in $V$ but not in $C$) and put it into $C$ until $C$ covering all of $V$. "$C$ covers all of $V$" means that each word in $V$ occurs at least once in a document in $C$.

More concretely, let $V_{\text{todo}}$ be the part of $V$ not covered by $C$, and let $V_{\text{done}}$ be $V - V_{\text{todo}}$. We iteratively put document $d$ into $C$ that maximizes $G(\cdot)$,

$$\begin{aligned} & G(d|\alpha, V_{\text{todo}}, V_{\text{done}}) \\ = \;& \alpha g(d|V_{\text{todo}}) + (1-\alpha) g(d|V_{\text{done}}), \end{aligned} \quad (1)$$

until $C$ covers all of $V$. We then define $g(\cdot)$ as

$$\begin{aligned} & g(d|V_x) \\ = \;& \frac{k_1+1}{k_1((1-b)+b\frac{|W(d)|}{E(|W(\cdot)|)})+1}|W(d)\cap V_x|, \end{aligned} \quad (2)$$

where $W(d)$ is the set of types in $d$, $E(|W(\cdot)|)$ is the average for $|W(\cdot)|$ over the whole corpus, and $k_1$ and $b$ are parameters that depend on the corpus. We set $k_1$ as 1.5 and $b$ as 0.75. $g(d|V_x)$ takes a large value when there is a large number of common types between $W(d)$ and $V_x$ and $d$ is short. These effects are due to $|W(d) \cap V_x|$ and $\frac{|W(d)|}{E(|W(\cdot)|)}$ respectively. As $g(\cdot)$ is based on the Okapi BM25 function (Robertson and Walker, 2000), which has been shown to be quite efficient in information retrieval,[9] we expected $g(\cdot)$ to be effective in retrieving documents relevant to the target vocabulary.

In Eq. (1), $\alpha$ is used to combine the scores of document $d$, which are obtained by using $V_{\text{todo}}$ and $V_{\text{done}}$. It is defined as

$$\alpha = \frac{|V_{\text{done}}|}{1+|V_{\text{done}}|} \quad (3)$$

This implies that even if $|W(d) \cap V_{\text{todo}}|$ is 1, it is as important as $|W(d) \cap V_{\text{done}}| = |V_{\text{done}}|$. Consequently, $G(\cdot)$ uses documents that have new types of the given vocabulary in preference to documents that have covered types.

To summarize, efficient courseware is constructed by putting document $d$ with maximum $G(\cdot)$ into $C$ until $C$ covers all of $V$. This allows us to construct efficient courseware because $G(\cdot)$ takes a large value when a document has a large number of new types and is short.

## 3 Experiment

This section describes how the courseware was constructed by applying the method described in the previous section. We will first describe the vocabulary and corpus used to construct the courseware and then present the statistics for the courseware.

### 3.1 Vocabulary

We used the specialized vocabulary used in the Test of English for International Communication (TOEIC) because it is one of the most popular English certification tests in Japan. The vocabulary was compiled by Chujo (2003) and Chujo et al. (2004), who confirmed that the vocabulary was useful in preparing for the TOEIC test. The vocabulary had 640 entries and we used 638 words from it that occurred at least once in the corpus as the target vocabulary.

### 3.2 Corpus

We used articles from English Wikipedia as the target corpus, which is a free-content encyclopedia that anyone can edit. The version we used in this study had 478,611 articles. From these, we first discarded stub and other non-normal articles. We also discarded short articles of less than 150 words. We then selected 60,498 articles that were referred to (linked) by more than 15 articles. This 15-link threshold was

[7] http://www.kotonoba.net/˜mutiyama/vocabridge/

[8] A *type* refers to a unique word, while a *token* refers to each occurrence of a type.

[9] BM25 and its variants have been proven to be quite efficient in information retrieval. Readers are referred to papers by the Text REtrieval Conference (TREC, http://trec.nist.gov/), for example.

set empirically to screen out noisy articles. Finally, we extracted a 150-word excerpt from the lead part of each of these 60,498 articles to prepare the target corpus. We set 150-word limit on an empirical basis to reduce the burden imposed on learners. In short, the target corpus consisted of 60,498 excerpts from the English Wikipedia. In the rest of the paper, we will use the term *an article* to refer to *an excerpt* that was extracted according to this procedure.

### 3.3 Example article

Figure 1 has an example of the articles in the courseware. It was the first article obtained with the algorithm. It shares 27 types and 49 tokens with the target vocabulary. These words are printed in **bold**.

**Corporate finance**

**Corporate finance** is the specific **area** of **finance dealing** with the **financial decisions corporations** make, and the tools and **analysis** used to make the **decisions**. The discipline as a whole may be divided between **long-term** and short-term **decisions** and techniques. Both share the same goal of enhancing **firm** value by **ensuring** that return on **capital exceeds cost** of **capital**. **Capital investment decisions** comprise the **long-term** choices about which **projects receive investment**, whether to **finance** that **investment** with equity or debt, and when or whether to pay dividends to shareholders. Short-term **corporate finance decisions** are called working **capital management** and **deal** with balance of **current** assets and **current** liabilities by **managing cash**, **inventories**, and short-term borrowing and lending (e.g., the **credit** terms **extended** to **customers**). **Corporate finance** is closely related to managerial **finance**, which is slightly broader in scope, describing the **financial** techniques **available** to all **forms** of business ... (more)

Figure 1: Example article

### 3.4 Courseware statistics

#### 3.4.1 Basic courseware statistics

Table 1 lists basic statistics for the courseware constructed from the target vocabulary and corpus.[10] The courseware consisted of 131 articles. Each article was 150 words long because only excerpts were used. The average number of tokens per article shared with the vocabulary ("num. of common tokens" in the Table) was 18.4 and that of types ("num. of common types") was 12.4. About $12.3\%(= \frac{18.4}{150} \times 100)$ of the tokens in each article were covered by the vocabulary. Each article in the courseware was referred to by 70.7 articles on average as can be seen from the bottom row. Table 1 indicates that articles in the courseware included many target words and were heavily referred to by other articles.

#### 3.4.2 Distribution of covered types

Figure 2 plots the increase in the number of covered types against the order (ranking) of articles that were put into the courseware. The horizontal axis represents the ranking of articles. The vertical axis indicates the number of covered types. The increase was sharpest when the ranking value was lowest (left of figure). The dotted horizontal lines indicate 50% and 90% of the target vocabulary. These lines cross the curved solid line at the 22nd and 83rd articles, i.e., 16.8% and 63.4% of the courseware, respectively. This means that learners can learn most of the target vocabulary from the beginning of the courseware. This is desirable because learners sometimes do not have enough time to read all the courseware.

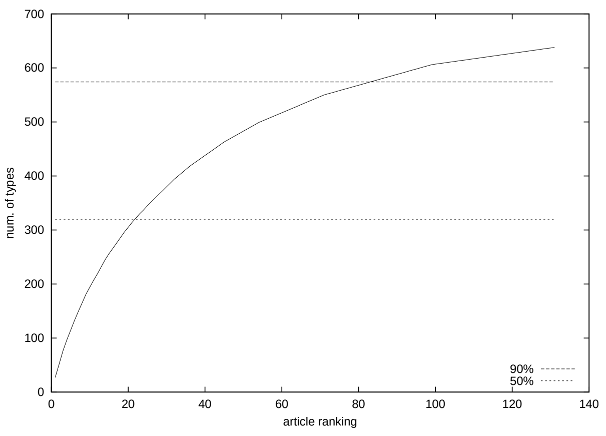


Figure 2: Increase in the number of covered types

#### 3.4.3 Document frequency distribution

Figure 3 has target words that occurred in eight articles or more. The numbers in parentheses indicate the document frequencies (DFs) of the words, where the *DF* of a word is the number of articles in which the word occurred. These words were the most basic words in the target vocabulary with respect to the courseware.

Table 2 lists the distribution of DFs. The first column lists the different DFs of the target words. The values in the "#DF" column are the numbers of

[10]On our web site, we prepared 10 sets of article sets called *course-1* to *course-10*. These 10 courses were obtained by repeatedly applying our algorithm to the English Wikipedia removing articles included in earlier courses. The statistics presented in this paper were calculated from the first courseware, *course-1*.

Table 1: Basic courseware statistics (number of articles: 131, length of each article: 150 words)

| | Average | SD | Min | Median | Max |
|---|---|---|---|---|---|
| Num. of common tokens | 18.4 | 10.8 | 1 | 16 | 55 |
| Num. of common types | 12.4 | 5.5 | 1 | 12 | 27 |
| Num. of incoming links | 70.7 | 145.3 | 16 | 32 | 1056 |

SD means standard deviation.

words that occurred in the corresponding DF articles. The "CUM" and "CUM%" columns show the cumulative numbers and percentages of words calculated from the values in the second column. As we can see from Table 2, more than 50% of the target words occurred in multiple articles. Consequently, learners were likely to be sufficiently exposed to efficiently learn the target vocabulary.

service (19), form (17), information (12), feature (12), operation (11), cost (11), individual (10), department (10), consumer (9), company (9), product (9), complete (9), range (9), law (9), associate (9), cause (9), consider (9), offer (9), provide (9), present (8), activity (8), due (8), area (8), bill (8), require (8), order (8)

Figure 3: Target words and their DFs.

Table 2: Document frequency distribution

| DF | #DF | CUM | CUM% |
|---|---|---|---|
| 19 | 1 | 1 | 0.2 |
| 17 | 1 | 2 | 0.3 |
| 12 | 2 | 4 | 0.6 |
| 11 | 2 | 6 | 0.9 |
| 10 | 2 | 8 | 1.3 |
| 9 | 11 | 19 | 3.0 |
| 8 | 7 | 26 | 4.1 |
| 7 | 20 | 46 | 7.2 |
| 6 | 25 | 71 | 11.1 |
| 5 | 35 | 106 | 16.6 |
| 4 | 36 | 142 | 22.3 |
| 3 | 71 | 213 | 33.4 |
| 2 | 118 | 331 | 51.9 |
| 1 | 307 | 638 | 100.0 |

## 4 Conclusion

While many teachers agree that vocabulary learning can be fostered by presenting words in context rather than isolating them from this, it is very difficult to prepare reading materials that contain the specialized vocabulary to be learned. We have proposed a method of automating this preparation process (Utiyama et al., 2004). We have found that our reading materials prepared from The Daily Yomiuri were effective in vocabulary learning (Tanimura and Utiyama, in preparation).

Our next goal is to distribute courseware (produced with our algorithm) to EFL teachers and learners so that we can receive wider feedback. To this end, we replaced The Daily Yomiuri, which is copyrighted, with the English Wikipedia, which is a free-content encyclopedia, and developed new courseware whose statistics were presented and discussed in this paper. This courseware, which is available on our web site, can be used to supplement classroom learning activities as well as self-study. We hope it will help EFL learners to learn and teachers to teach a broader range of vocabulary.

## References

K. Chujo, T. Ushida, A. Yamazaki, M. Genung, A. Uchibori, and C. Nishigaki. 2004. Bijuaru beishikku niyoru TOEIC-yoo goiryoku yoosei sofutowuea no shisaku (3) [The development of English CD-ROM material to teach vocabulary for the TOEIC test (utilizing Visual Basic): Part 3]. Journal of the College of Industrial Technology, Nihon University, 37, 29-43.

K. Chujo. 2003. Eigo shokyuushamuke TOEIC Goi 1 & 2 no sentei to sono kouka [Selecting TOEIC vocabulary 1 & 2 for beginning-level students and measuring its effect on a sample TOEIC test]. Journal of the College of Industrial Technology Nihon University, 36: 27-42.

S. E. Robertson and S. Walker. 2000. Okapi/Keenbow at TREC-8. In *Proc. of TREC 8*, pages 151–162.

Midori Tanimura and Masao Utiyama. in preparation. Reading materials for learning TOEIC vocabulary based on corpus data.

Masao Utiyama, Midori Tanimura, and Hitoshi Isahara. 2004. Constructing English reading courseware. In *PACLIC-18*, pages 173–179.

# Reformatting Web Documents via Header Trees

**Minoru Yoshida and Hiroshi Nakagawa**
Information Technology Center, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
CREST, JST
mino@r.dl.itc.u-tokyo.ac.jp, nakagawa@dl.itc.u-tokyo.ac.jp

## Abstract

We propose a new method for reformatting web documents by extracting semantic structures from web pages. Our approach is to extract trees that describe hierarchical relations in documents. We developed an algorithm for this task by employing the EM algorithm and clustering techniques. Preliminary experiments showed that our approach was more effective than baseline methods.

## 1 Introduction

This paper proposes a novel method for reformatting (i.e., changing visual representations,) of web documents. Our final goal is to implement the system that appropriately reformats layouts of web documents by separating semantic aspects (like XML) from layout aspects (like CSS) of web documents, and changing the layout aspects while retaining the semantic aspects.

We propose a *header tree*, which is a reasonable choice as a semantic representation of web documents for this goal. Header trees can be seen as variants of XML trees where each internal node is not an XML tag, but *a header* which is a part of document that can be regarded as tags annotated to other parts of the document. Titles, headlines, and attributes are examples of headers. The left part of Figure 1 shows an example web document. In this document, the headers are `About Me`, which is a title, and `NAME` and `AGE`, which are attributes. (For example, `NAME` can be seen as a tag annotated to `John Smith`.) Figure 2 shows a header tree for the example document. It should be noted that each node is labeled with parts of HTML pages, not abstract categories such as XML tags.

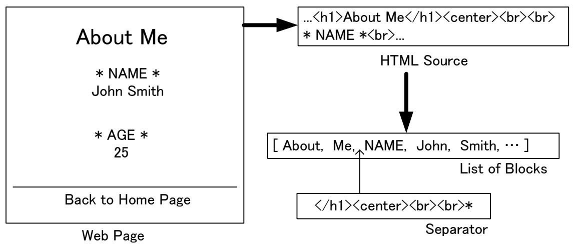


Figure 1: An Example Web Document and Conversion from HTML Documents to Block Lists.

Therefore, the required task is to extract header trees from given web documents. Web documents can be reformatted by converting their header trees into various forms including Powerpoint-like indented lists, HTML tables[1], and Tree-class objects of Java. We implemented the system that produces these representations by extracting header trees from given web documents.

One application of such reformatting is a *web browser on small devices* that shows extracted header trees regardless of original HTML visual rendering. Trees can be used as compact representations of web documents because they show internal structures of web documents concisely, and they can be further augmented with open/close operations on each node for the purpose of closing unnecessary nodes, or sentence summarization on leaf nodes containing long sentences. Another application is a *layout changer*, which change a layout (i.e., HTML tag usage) of one web page to another, by aligning extracted header trees of two web documents. Other applications include HTML to XML transformation and audio-browsable web content (Mukherjee et al., 2003).

[1]For example, the first column represents the root, the second column represents its children, etc.

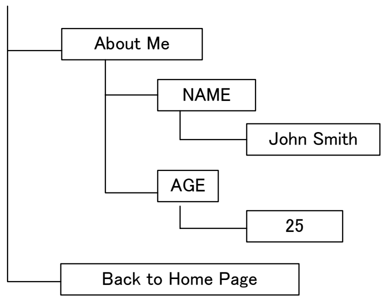


Figure 2: A Header Tree for the Example Web Document

### 1.1 Related Work

Several studies have addressed the problem of extracting logical structures from general HTML documents without labeled training examples. One of these studies used domain-specific knowledge to extract information used to organize logical structures (Chung et al., 2002). However, their approach cannot be applied to domains for which any knowledge is not provided. Another type of study employed algorithms to detect *repeated patterns* in a list of HTML tags and texts (Yang and Zhang, 2001; Nanno et al., 2003), or more structured forms (Mukherjee et al., 2003; Crescenzi et al., 2001; Chang and Lui, 2001) such as DOM trees. This approach might be useful for certain types of web documents, particularly those with highly regular formats such as `www.yahoo.com` and `www.amazon.com`. However, in many cases, HTML tag usage does not have so much regularity, and, there are even the case where headers do not repeat at all. Therefore, this type of algorithm may be inadequate for the task of header extraction from arbitrary web documents.

The remainder of this paper is organized as follows. Section 2 defines the terms used in this paper. Section 3 provides the details of our algorithm. Section 4 lists the experimental results and Section 5 concludes this paper.

## 2 Definitions

### 2.1 Definition of Terms

Our system decomposes an HTML document into a list of *blocks*. A block is defined as the part of a web document that is separated by a *separator*. A separator is a sequence of HTML tags and *symbols*. Symbols are defined as characters in texts that are neither numbers nor letters. Figure 1 shows an example of the conversion of an HTML document to a list of blocks.

[ [About Me, [NAME, John Smith], [AGE, 25] ], Back to Home Page] ]

Figure 3: A List Representation of the Example Web Document

A header is defined as a block that modifies subsequent blocks. In other words, a block that can be a tag annotated to subsequent blocks is defined as a header. Some examples of headers are Titles (e.g., “About Me”), Headlines (e.g., “Here is my profile:”), Attributes (e.g., “Name”, “Age”, etc.), and Dates.

### 2.2 Definition of the Task

The system produces header trees for given web documents. A header tree can be seen as an indented list of blocks where the level of each node’s indent is equal to the depth of the node, as shown in Figure 2. Therefore, the main part of our task is to give a *depth* to each block in a given web document. After that, some heuristic rules are employed to construct header trees from a list of depths. In the next section, we discuss the task of assigning a depth to each block. Therefore, an input to the system is a list of blocks and the output is a list of depths.

The system also produces *nested-list representation* of header trees for the purpose of evaluation. In nested-list representation, each node that has children is represented by the list whose first element represents the parent and remaining elements represent the children. Figure 3 shows list representation of the tree in Figure 2.

## 3 Header Extraction Algorithm

In this section, we describe our algorithm that receives a list of blocks and returns a list of depths.

### 3.1 Basic Concepts

The algorithm proceeds in two steps: separator categorization and block clustering. The first step estimates local block relations (i.e., relations between neighboring blocks) via probabilistic models for characters and tags that appear around separators. The second step supplements the first by extracting the undetermined relations between blocks by focusing on global features, i.e., regularities in HTML tag sequences. We employed a *clustering* framework to implement a flexible regularity detection system that is robust to noise.

### 3.2 STEP 1: Separator Categorization

The algorithm classifies each block relation into one of three classes: NON-BOUNDARY, RELATING,

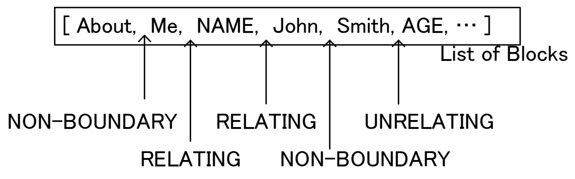


Figure 4: An Example of Separator Categorization.

and UNRELATING. Both RELATING and UNRELATING can be considered to be boundaries; however, blocks that sandwich RELATING separators are regarded to consist of a header and its modified block. Figure 4 shows an example of separator categorization for the list of blocks in Figure 1.

The left block of a RELATING separator must be in the smaller depth than the right block. Figure 2 shows an example. In this tree, `NAME` is in a smaller depth than `John`. On the other hand, both the left and right blocks in a NON-BOUNDARY separator must be in the same depth in a tree representation, for example, `John` and `Smith` in Figure 2.

### 3.2.1 Local Model

We use a probabilistic model that assumes the locality of relations among separators and blocks. In this model, each separator $s$ and the strings around it, $l$ and $r$, are modeled by means of the hidden variable $c$, which indicates the class in which $s$ is categorized. We use the character zerogram, unigram, or bigram (changed according to the number of appearances[2]) for $l$ and $r$ to avoid data sparseness problems.

For example, let us consider the following part of the example document:

`NAME: John Smith.`

In this case, `:` is a separator, `ME` is the left string and `Jo` is the right string.

Assuming the locality of separator appearances, the model for all separators in a given document set is defined as $P(\mathbf{l},\mathbf{s},\mathbf{r}) = \prod P(l,s,r)$ where $\mathbf{l}$ is a vector of left strings, $\mathbf{s}$ is a vector of separators, and $\mathbf{r}$ is a vector of right strings.

The joint probability of obtaining $l$, $s$, and $r$ is

$$P(l,s,r) \;=\; P(s)P(c|s)P(l|c)P(r|c)$$

assuming that $l$ and $r$ depend only on $c$: a class of relation between the blocks around $s$.[3][4]

[2]This generalization is performed by a heuristic algorithm. The main idea is to use a bigram if its number of appearances is over a threshold, and unigrams or zerograms otherwise.

[3]If the frequency for $(l,r)$ is over a threthold, $P(l,r|c)$ is used instead of $P(l|c)P(r|c)$.

[4]If the frequency for $s$ is under a threthold, $s$ is replaced by its longest prefix whose frequency is over the threthold.

Based on this model, each class of separators is determined as follows:

$$\hat{c} = \arg\max_{c} P(s)P(c|s)P(l|c)P(r|c).$$

The hidden parameters $P(c|s)$, $P(l|c)$, and $P(r|c)$, are estimated by the EM algorithm (Dempster et al., 1977). Starting with arbitrary initial parameters, the EM algorithm iterates E-STEPs and M-STEPs in order to increase the (log-)likelihood function $\log P(\mathbf{l},\mathbf{s},\mathbf{r}) = \sum \log P(l,s,r)$.

To characterize each class of separators, we use a set of typical symbols and HTML tags, called *representatives* from each class. This constraint contributes to give a structure to the parameter space.

### 3.3 STEP 2: Block Clustering

The purpose of block clustering is to take advantage of the regularity in visual representations. For example, we can observe regularity between `NAME` and `AGE` in Figure 1 because both are sandwiched by the character $\star$ and preceded by a null line. This visual representation is described in the HTML source as, for example,

```
... <br><br>⋆ NAME ⋆<br> ...
... <br><br>⋆ AGE ⋆<br> ...
```

Our idea is to define the similarities between (context of) blocks based on the similarities between their surrounding separators. Each separator is represented by the vector that consist of symbols and HTML tags included in it, and the similarity between separators are calculated as cosine values. The algorithm proceeds in a bottom-up manner by examining a given block list from tail to head, finding the block that is the most similar to the current block, and collecting them into the same cluster. After that, all blocks in the same cluster is assigned the same depth.

## 4 Preliminary Experiments

We used a training data that consists of 1,418 web documents[5] of moderate file size[6] that did not have “src” or “script” tags[7]. The former criteria is based on the observation that too small or too large documents are hard to use for measuring performance of algorithms, and the latter criteria is caused by the fact our system currently has no module to handle image files as blocks.

We randomly selected 20 documents as test documents. Each test document was bracketed by hand

[5]They are collected by retrieving all user pages on one server of a Japanese ISP.

[6]from 1,000 to 10,000 bytes

[7]Src tags indicate inclusion of image files, java codes, etc

| Algorithm | Recall | Precision | F-measure |
|---|---|---|---|
| OUR ALGORITHM | 0.477 | 0.266 | 0.329 |
| NO-CL | 0.178 | 0.119 | 0.139 |
| NO-EM | 0.389 | 0.211 | 0.265 |
| PREV | 0.144 | 0.615 | 0.202 |

Table 1: Macro-Averaged Recall, Precision, and F-measure on Test Documents

to evaluate machine-made bracketings. The performance of web-page structuring algorithms can be evaluated via the nested-list form of tree by *bracketed recall* and *bracketed precision* (Goodman, 1996). Recall is the rate that bracketing given by hand are also given by machine, and precision is the rate that bracketing given by machine are also given by hand. F-measure is a harmonic mean of recall and precision that is used as a combined measure. Recall and precision were evaluated for each test document and they were averaged across all test documents. These averaged values are called *macro-average recall, precision, and f-measure* (Yang, 1999).

We implemented our algorithm and the following three ones as baselines.

**NO-CL** does not perform block clustering.

**NO-EM** does not perform the EM-parameter-estimation. Every boundary but representatives is defined to be categorized as "UNRELATING".

**PREV** performs neither the EM-learning nor the block clustering. Every boundary but representatives is defined to be categorized as "NON-BOUNDARY"[8]. It uses the heuristics that "every block depends on its previous block."

Table 1 shows the result. We observed that use of both the EM-learning and block clustering resulted in the best performance. NO-EM performs the best among the three baselines. It suggests that only relying on HTML tag information is not a so bad strategy when the EM-training is not available because of, for example, the lack of a sufficient number of training examples.

Results on the documents that were rich in HTML tags with highly coherent layouts were better than those on the others like the documents with poor separators such as only one space character or one line feed. Some of the current results on the documents with such poor visual cues seemed difficult for use in practical systems, which indicates our system still leaves room for improvement.

[8]This strategy is based on the fact that it maximized the performance in a preliminary investigation.

## 5 Conclusions and Future Work

This paper proposed a method for reformatting web documents by extracting header trees that give hierarchical structures of web documents. Preliminary experiments showed that the proposed algorithm was effective compared with some baseline methods. However, the performance of the algorithm on some of the test documents was not sufficient for practical use. We plan to improve the performance by, for example, using larger amount of training examples. Finding other reformatting strategies in addition to the ones proposed in this paper is also important future work.

## References

Chia-Hui Chang and Shao-Chen Lui. 2001. IEPAD: Information extraction based on pattern discovery. In *Proceedings of WWW2001*, pages 681–688.

Christina Yip Chung, Michael Gertz, and Neel Sundaresan. 2002. Reverse engineering for web data: From visual to semantic structures. In *ICDE*.

Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. 2001. ROADRUNNER: Towards automatic data extraction from large web sites. In *Proceedings of VLDB '01*, pages 109–118.

A.P. Dempster, N.M. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society: Series B*, 39:1–38.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of ACL96*, pages 177–183.

Saikat Mukherjee, Guizhen Yang, Wenfang Tan, and I.V. Ramakrishnan. 2003. Automatic discovery of semantic structures in HTML documents. In *Proceedings of ICDAR 2003*.

Tomoyuki Nanno, Suguru Saito, and Manabu Okumura. 2003. Structuring web pages based on repetition of elements. In *Proceedings of WDA2003*.

Yudong Yang and Hongjiang Zhang. 2001. HTML page analysis based on visual cues. In *Proceedings of ICDAR01*.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *INRT*, 1:69–90.

# Author Index