

NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

NATURAL LANGUAGE GENERATION

Title Generation for Machine-Translated Documents

Rong Jin

Language Technologies Institute
Carnegie Mellon University,
Pittsburgh, PA, 15213
Rong+@cs.cmu.edu

Alexander G. Hauptmann

Dept. of Computer Science
Carnegie Mellon University,
Pittsburgh, PA, 15213
Alex+@cs.cmu.edu

Abstract

In this paper, we present and compare automatically generated titles for machine-translated documents using several different statistics-based methods. A Naïve Bayesian, a K-Nearest Neighbour, a TF-IDF and an iterative Expectation-Maximization method for title generation were applied to 1000 original English news documents and again to the same documents translated from English into Portuguese, French or German and back to English using SYSTRAN. The AutoSummarization function of Microsoft Word was used as a base line. Results on several metrics show that the statistics-based methods of title generation for machine-translated documents are fairly language independent and title generation is possible at a level approaching the accuracy of titles generated for the original English documents.

1. Introduction

Before we discuss generating target language titles for documents in a different source language, let's consider in general the complex task of creating a title for a document: One has to understand what the document is about, one has to know what is characteristic of this document with respect to other documents, one has to know how a good title sounds to catch attention and how to distill the essence of the document into a title of just a few words. To generate a title for a machine-translated document becomes even more challenging because we have to deal with syntactic and semantic translation errors generated by the automatic translation system.

Generating text titles for machine translated documents is very worthwhile because it produces a very compact target language representation of the original foreign language document, which will help readers to understand the important information contained in the document quickly, without requiring a translation of the complete document. From the viewpoint of machine learning, studies on how well general title generation methods can be adapted to errorful machine translated documents and

which methods perform better than others will be very helpful towards general understanding on how to discover knowledge from errorful or corrupted data and how to apply learned knowledge in 'noisy' environments.

Historically, the task of title generation is strongly connected to more traditional document summarization tasks (Goldstein *et al.*, 1999) because title generation can be thought of as extremely short summarization. Traditional summarization has emphasized the extractive approach, using selected sentences or paragraphs from a document to provide a summary (Strzalkowski *et al.*, 1998, Salton *et al.*, 1997, Mitra *et al.*, 1997). Most notably, McKeown *et al.* (1995) have developed systems that extract phrases and recombine elements of phrases into titles.

More recently, some researchers have moved toward "learning approaches" that take advantage of training data (Witbrock and Mittal, 1999). Their key idea was to estimate the probability of generating a particular title word given a word in the document. In their approach, they ignore all document words that do not appear in the title. Only document words that effectively *reappear* in the title of a document are counted when they estimate the probability of generating a title word wt given a document word wd as: $P(wt|wd)$ where $wt = wd$. While the Witbrock/Mittal Naïve Bayesian approach is not in principle limited to this constraint, our experiments show that it is a very useful restriction. Kennedy and Hauptmann (2000) explored a generative approach with an iterative Expectation-Maximization algorithm using most of the document vocabulary. Jin and Hauptmann (2000a) extended this research with a comparison of several statistics-based title word selection methods.

In our approach to the title generation problem we will assume the following:

First, the system will be given a set of target language training data. Each datum consists of a document and its corresponding title. After exposure to the training corpus, the system should be able to generate a title for any unseen document. All source language documents will be translated into the target language before any title generation is attempted.

We decompose the title generation problem into two phases:

- **learning and analysis** of the training corpus and
- **generating a sequence of words** using learned statistics to form the title for a new document.

For **learning and analysis** of the training corpus, we present five different learning methods for comparison. All the approaches are described in detail in section 2.

- A Naïve Bayesian approach with limited vocabulary. This closely mirrors the experiments reported by Witbrock and Mittal (1999).
- A Naïve Bayesian approach with full vocabulary. Here, we compute the probability of generating a title word given a document word for all words in the training data, not just those document words that reappear on the titles.
- A KNN (k nearest neighbors) approach, which treats title generation as a special classification problem. We consider the titles in the training corpus as a fixed set of labels, and the task of generating a title for a new document is essentially the same as selecting an appropriate label (i.e. title) from the fixed set of training labels. The task reduces to finding the document in the training corpus, which is most similar to the current document to be titled. Standard document similarity vectors can be used. The new document title will be set to the title for the training document most similar to the current document.
- Iterative Expectation-Maximization approach. This duplicates the experiments reported by Kennedy and Hauptmann (2000).
- Term frequency and inverse document frequency (TFIDF) method (Salton and Buckley, 1988). TFIDF is a popular method used by the Information Retrieval community for measuring the importance of a term related to a document. We use the TFIDF score of a word as a measurement of the potential that this document word will be adopted into the title, since important words have higher chance of being used in the title.

For the **title-generating** phase, we can further decompose the issues involved as follows:

- Choosing appropriate title words. This is done by applying the learned knowledge from one of the six methods examined in this paper.
- Deciding how many title words are appropriate for this document title. In our experiments we simply fixed the length of generated titles to the average expected title length for all comparisons.
- Finding the correct sequence of title words that forms a readable title ‘sentence’. This was done by applying a language model of title word tri-

grams to order the newly generated title word candidates into a linear sequence.

Finally, as a baseline, we also used the extractive summarization approach implemented as *AutoSummarize* in Microsoft Word that selects the “best” sentence from the document as a title.

The outline of this paper is as follows: This section gave an introduction to the title generation problem. Details of our approach and experiments are presented in Section 2. The results and their analysis are presented in Section 3. Section 4 discusses our conclusions drawn from the experiment and suggests possible improvements.

2. Title Generation Experiment across Multiple Languages.

We will first describe the data used in our experiments, and then explain and justify our evaluation metrics. The six learning approaches will then be described in more detail at the end of this section.

2.1 Data Description

The experimental dataset comes from a CD of 1997 broadcast news transcriptions published by Primary Source Media (1997). There were a total of roughly 50,000 news documents and corresponding titles in the dataset. The training dataset was formed by randomly picking four documents-title pairs from every five pairs in the original dataset. The size of training corpus was therefore 40,000 documents and their titles. We fixed the size of the test collection at 1,000 items from the unused document-title pairs. By separating training data and test data in this way, we ensure strong overlap in topic coverage between the training and test datasets, which gives the learning algorithms a chance to play a bigger role.

Since we did not have a large set of documents with titles in multiple parallel languages, we approximated this by creating machine-translated documents from the original 1000 training documents with titles as follows:

Each of the 1000 test documents was submitted to the SYSTRAN machine translation system (<http://babelfish.altavista.com>) and translated into French. The French translation was again submitted to the SYSTRAN translation system and translated back into English. This final retranslation resulted in our French machine translation data. The procedure was repeated on the original documents for translation into Portuguese and German to obtain two more machine translated sets of identical documents. For all languages, the average vocabulary overlap between the translated documents and the original documents was around 70%. This implies that at least 30% of the English words were translated incorrectly during translation from English to a foreign language and back to English.

2.2 Evaluation Metric

In this paper, two evaluations are used, one to measure **selection quality** and another to measure **accuracy of the sequential ordering** of the title words.

To measure the **selection quality** of title words, an F1 metric was used (Van Rjiesbergen, 1979). For an automatically generated title T_{auto} , F1 is measured against the correspondent human assigned title T_{human} as follows:

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Precision and recall is measured as the number of identical words in T_{auto} and T_{human} over the number of words in T_{auto} and the number of words in T_{human} respectively. Since we are focusing here on the choice of appropriate title words, F1 is the most appropriate measure for this purpose. Obviously the sequential word order of the generated title words is ignored by this metric. However, preliminary tests with human ratings for the automatically generated titles suggest a strong correlation between F1 and human quality judgments.

Accuracy of the sequential ordering: To measure how well a generated title compared to the original human generated title in terms of word order, we measured the number of correct title words in the hypothesis titles that were in the same order as in the reference titles using the dynamic alignment algorithm described by (Nye, 1984).

To make all approaches comparable (except MS Word AutoSummarize and KNN), only 6 title words were generated by each method, as 6 was the average number of title words in the training corpus. Since AutoSummarize in Microsoft Word selects a complete sentence from the test document as the title, the restriction to a title length of exactly 6 words often prevents AutoSummarize from producing any title sentence. Thus, we allow longer titles for AutoSummarize in Microsoft Word. The KNN method always uses the complete title of the document in the training corpus most similar to the test document as the title for the test document and thus the restriction of six words does not apply to titles generated by KNN. Since we wanted to emphasize content word accuracy, stop words were removed throughout the training and testing documents and titles.

2.3 Description of Title Generation Approaches

As we mentioned in the introduction, we compared five statistics-based title generation methods together with the baseline “extractive” approach. They were:

1. **Naïve Bayesian** approach with **limited vocabulary** (NBL). Essentially, this algorithm duplicates the work by Witbrock and Mittal (1999), which tries to capture the correlation between the words in the document and the words in the title. For each title word TW, it counts the occurrences of document word DW, if DW is the same as TW (i.e. $DW = TW$). To generate a title, we merely apply the statistics to the test documents for generating titles and select the top title words TW where $P(TW | DW)$ is largest.
2. **Naïve Bayesian** approach with **full vocabulary** (NBF). The previous approach counts only the cases where the title word and the document word are the same. This restriction is based on the assumption that a document word is only able to generate a title word with same surface string. The constraint can be easily relaxed by counting all the document-word-title-word pairs and apply this full statistics on generating titles for the test documents. In all other respects, this approach is the same as the previous one.
3. **K nearest neighbor** approach (KNN). This algorithm is similar to the KNN algorithm applied to topic classification in (Yang *et al*, 1994). It treats the titles in the training corpus as a set of fixed labels. For each new document, instead of creating a new title, it tries to find an appropriate “label”, which is equivalent to searching the training document set for the closest related document. This training document title is then used for the new document. In our experiment, we use SMART (Salton, 1971) to index our training documents and test documents with the weight schema “ATC”. The similarity between documents is defined as the dot product between document vectors. The training document closest related to the test document is found by computing the similarity between the test document and each training document ($K=1$).
4. **Iterative Expectation-Maximization** approach (EM). This algorithm reproduces the work by Kennedy and Hauptmann (2000), which treats title generation as a translation problem. We view a document as written in a ‘verbose’ language and its corresponding title as written in a ‘concise’ language. The approach builds a translation model (Brown *et al.*, 1990), between verbose and concise languages, based on the documents and titles in the training corpus and applies the learned translation model to generate titles for new documents. The essential difference between this approach and Naïve Bayesian approaches is that EM treats the title word generation probability given a document as the sum of the title word generation probability from all the document words while the Naïve Bayesian approach treats it as the product of the title word generation probability from all document words.
5. **Term frequency and inverse document frequency approach** (TF.IDF). Term frequency TF, i.e. the frequency of words occurring in a document, shows how important a word is inside a document. Inverse document frequency IDF, i.e. the log of the total number of documents divided by the number of documents containing this word, shows how rarely a term appears in the collection. The product of these

two factors, i.e. TF.IDF, gives the importance of a term related to a document (Salton and Buckley, 1988). The highest-ranking TF.IDF document words are chosen for the title word candidates. The inverse document frequency for each term is computed based on how often it appears in the training corpus.

6. **Extractive summarization** approach (AUTO). We use the AutoSummarize function built into Microsoft Word as a demonstration of an extractive approach, which select the “best” sentence from the document as the title.

2.4 The Sequencing Process for Title Word Candidates

To generate an ordered, linear set of candidates, equivalent to what we would expect to read from left to right, we built a statistical trigram language model using the CMU-Cambridge Spoken Language Modeling toolkit (Clarkson and Rosenfeld, 1997) and the 40,000 titles in the training set. This language model was used to determine the most likely order of the title word candidates generated by the NBL, NBF, EM and TF.IDF methods. The KNN and AUTO generated titles were already in natural sequence.

3 Experimental Results and Discussion

To illustrate the quality of the results, we first show an example of machine-generated titles and then present quantitative results and their analysis.

3.1 Example

The following is an excerpt of a document translated from English to Portuguese and back to English together with its original, human-assigned English title. The translation shows all the typical characteristics and problems of machine-translated documents. The corresponding machine-generated titles are shown in Table 1.

Original Title: O.J. SIMPSON CIVIL TRIAL - CASE GOES TO JURY SOON

Document:

... THE CIVIL EXPERIMENTATION DE SIMPSON OF THE J WILL BE SOON IN THE HANDS OF THE JURY. THE LAWYERS FOR BOTH THE SIDES GIVE TO ITS ADDITIONS THE ADVANCED FOLLOWING WEEK AND THEN THE DELIBERATIONS START. THE JURY IN THE CIVIL EXPERIMENTATION IS JANE CLAYSON OF B C. HERE. DE SIMPSON OF THE J IS IS OF THE CUT TODAY WHEN THE JUDGE AND THE LAWYERS IN BOTH THE SIDES TO WORK FOR ARE OF INSTRUCTIONS JURY. THE PLAINTIFFS HAD FINISHED ITS CASE IT REBUTTAL WITH SOME HARMFUL TESTIMONY OF A CONNOISSEUR WHO OF THE PHOTOGRAPH OMS AUTHENTICATED THIRTY PICTURES RECENTLY DISCOVERED DE SIMPSON IT J THAT CONSUMES LOW SHOES THE

SAME DE BRUNO MAGLI STYLE RARE THE ASSASSIN CONSUMED. THE THEORIES OF THE CONTAMINATION OF THE DEFENSE ARE CITAÇÕES ABSOLUTELY RIDICULOUS OF CITATIONS. THE ARGUMENTS DE F THEY ARE PROGRAMADOS STILL TO START IN TUESDAY. THIS JURY COULD START THE CASE AND START TO DELIBERATE IN THURSDAY. ...

Method	Title
NBL	white house simpson civil case jury
NBF	Continuing coverage simpson civil trial president
EM	Continuing coverage simpson civil trial jury
AUTO	civil experimentation de the arguments de f
KNN	oj simpson civil trial
TF.IDF	simpson jury start de jane additions

Table 1 shows the machine-generated titles for a document translated from English to Portuguese and back to English.

3.2 Results and Discussion

We compared the machine-generated titles against reference titles using both F1 metrics and number of correct title words in the correct order. Figure 1 and 2 show the F1 scores and the average number of correct title words in the correct order for each method over both original documents and translated documents.

Performance of learning methods is relatively language independent. According to both the metrics of F1 and the average number of title words in the correct order, the performance for documents translated from English to French and Portuguese and back to English is quite similar for all six different methods. Performance for documents translated from English to German and back to English is somewhat worse than for French and Portuguese. This may be due to the underlying SYSTRAN translation system, or inherent in the inflectional and noun-compounding characteristics of German which distinguish it from the other languages examined here.

AutoSummarization in Microsoft Word performs poorly. In terms of F1 and the average number of title words in the correct word, AutoSummarization from Microsoft Word performs much worse than the methods KNN, TF.IDF, NBL and EM over either the original documents or translated documents. The only exception is that AutoSummarization appears to work fine for the original documents in terms of the average number of title words in the correct order. We believe this is due to the fact that AutoSummarization is allowed a much longer title length, which increases the chance to catch the right title word and improves the average number of title words in the correct order. This confirms other re-

search, which found that extractive summarization will only work well if there is much redundancy in the data and the summarization is much greater than 10% of the document size (Mittal *et al* 1999). Furthermore, the extraction-based approach is unable to take full advantage of the training corpus.

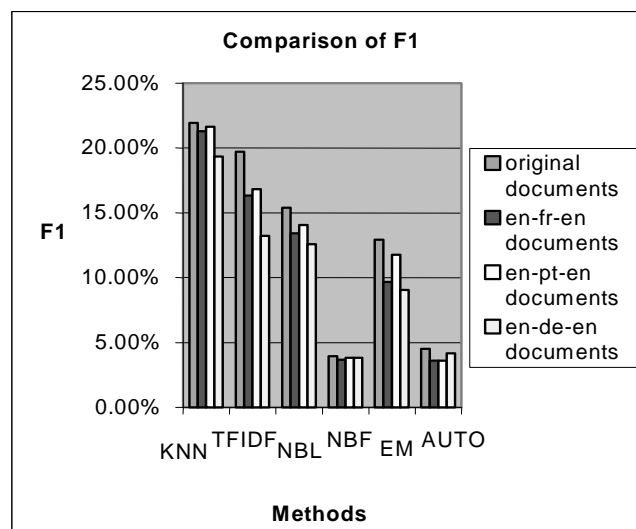


Figure 1 shows the F1 scores for the machine-generated titles. For each method, there are four bars representing the F1 scores for the titles generated from the original documents and the translated documents. The legends en-fr-en, en-pt-en and en-de-en represent the documents translated from English document to French, Portuguese and German and back in English, respectively.

K-Nearest Neighbor (KNN) performs extremely well. For both the original documents and the translated documents, KNN performs better than the other methods according to both the metrics of F1 and the average number of title words in the correct order. KNN works well here because the training and test sets were constructed to guarantee good overlap in content coverage. Even though our second best method, TF.IDF, shows performance close to KNN for the original documents, it degrades much more than KNN on all the three sets of machine-translated documents. There is almost no degradation for KNN over the documents translated from English to French and Portuguese and back to English. The large degradation for KNN over the document translated from English to German and back to English may be attributed to the fact that German is a quite different language from English as was already discussed. Actually, Jin and Hauptmann (2000b) have shown that KNN is also resilient to corruption from automatic speech recognition in generating titles for speech recognized documents. Thus, we conclude that KNN is a good approach for automatic title generation because of its simplicity and robustness to corrupt data.

Naive Bayesian with limit vocabulary (NBL) performs much better than Naive Bayesian with full vocabulary (NBF). The difference between NBF and NBL is that NBL assumes a document word can only generate a title word with the same surface string. This very strong assumption discards information about a lot of words. However, the results tell us that some information can be safely ignored. In NBF, nothing distinguishes between important words and trivial words, and the co-occurrence between all document words and title words is measured equally. This lets frequent, but unimportant words dominate the document-word-title-word correlation. As an extreme example, stop words show up frequently in every document. However, they have little effect on choosing title words. Thus, even though NBF seems to exploit more knowledge than NBL, it introduces more noise by not limiting the effects of frequent, but unimportant words.

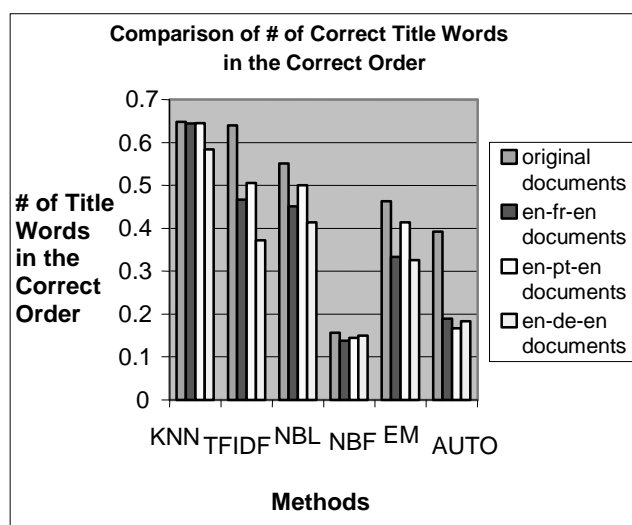


Figure 2 shows the average number of correct title words in the correct order. For each method, there are four bars representing the average number of correct title words in the correct order for the titles generated from the original documents and the translated documents. The legends en-fr-en, en-pt-en and en-de-en represent the documents translated from English document to French, Portuguese and German and back into English, respectively.

TF.IDF performs surprisingly well compared to the other true learning approaches. Surprisingly, the 'heavy' learning approaches, which take full advantage of the training corpus, such as Naive Bayesian with limit vocabulary (NBL), Naive Bayesian with full vocabulary (NBF) and Expectation-Maximization (EM) didn't outperform the shallow learning approach TF.IDF. Even though NBL, NBF and EM try to learn the association between title words and document words, they show no advantage over the simple TF.IDF approach, which selects the title words from the document based on the TF.IDF score, without learning anything about the titles. One suspicion is that learning the association between

document words and title words by directly inspecting the document and its title is very problematic. Many words in a document don't reflect its content. For example, in many documents there are extraneous paragraphs and copyright notices. Those word occurrences will blur the statistics and mislead the title word selection. A better strategy may be to first distill the document into essential content words and then compute the association between the distilled documents and their titles.

4 Conclusion

While title generation is far from a solved problem in one language, in this research we have, for the first time, applied learning approaches to title generation across languages. The research results show that automatic title generation is feasible on foreign language documents, despite gross errors in machine translation. Due to the flexibility and human readability issues of titles, the automatic evaluation metrics may not be able to reflect correctly the quality of titles. Thus, more work is needed to determine the human readability of the automatically generated titles, as well as the consistency between automatic evaluation metrics and human judgment. A validation with real cross-lingual documents is also desirable. In real life you would want to translate from French documents to English titles. You would train on English documents with English titles, and test on French documents translated into English.

References

- [Goldstein *et al.*, 1999] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing, Text Documents: Sentence Selection and Evaluation Metrics, *Proceedings of SIGIR 99*, Berkeley, CA, August 1999.
- [Strzalkowski *et al.*, 1998] T. Strzalkowski, J. Wang, and B. Wise, A robust practical text summarization system, *AAAI Intelligent Text Summarization Workshop*, pages 26--30, Stanford, CA, March 1998.
- [Salton *et al.*, 1997] G. Salton, A. Singhal, M. Mitra, and C. Buckley, Automatic text structuring and summary, *Info. Proc. And Management*, 33(2): 193-207, March 1997.
- [Mitra *et al.*, 1997] M. Mitra, A. Singhal, and C. Buckley, Automatic text summarization by paragraph extraction, *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain.
- [McKeown *et al.*, 1995] K. McKeown, J. Robin and K. Kukich, Generating Concise Natural Language Summaries, *Information Processing and Management*, 31 (5), pp.703-733, 1995.
- [Witbrock and Mittal, 1999] M. Witbrock and V. Mittal, Ultra-Summarization: A Statistical Approach to Generating Highly Condensed Non-Extractive Summaries, *Proceedings of SIGIR 99*, Berkeley, CA, August 1999
- [Kennedy and Hauptmann, 2000] P. Kennedy and A.G. Hauptmann, Automatic Title Generation for the Informedia Multimedia Digital Library, *ACM Digital Libraries, DL-2000*, San Antonio Texas, May 2000, in press.
- [Salton and Buckley, 1988] G. Salton and C. Buckley, Term-weighting approaches in automatic text retrieval, *Information Processing and Management*, 24, 513—523, 1988
- [Yang and Chute, 1994] Y. Yang and C.G. Chute, An example-based mapping method for text classification and retrieval, *ACM Transactions on Information Systems (TOIS)*, 12(3): 252-77. 1994.
- [Van Rjiesbergen, 1979] Van Rjiesbergen. Butterworths, *Information Retrieval*, Chapter 7. London, 1979.
- [Salton, 1971] G. Salton, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice Hall, Englewood Cliffs, New Jersey. 1971.
- [Clarkson and Rosenfeld, 1997] P.R. Clarkson and R. Rosenfeld. *Statistical Language Modeling Using the CMU-Cambridge Toolkit* Proceedings ESCA Eurospeech 1997
- [Mittal, *et al.*, 1999] V. Mittal, M. Kantrowitz, J. Goldstein and J. Carbonell, Selecting Text Spans for Document Summaries: Heuristics and Metrics, *AAAI-99*, 1999.
- [Broadcast News, 1997] Primary Source Media, Broadcast News CDRom, Woodbridge, CT, 1997
- [Nye, 1984] H. Nye, The Use of a One Stage Dynamic Programming Algorithm for Connected Word Recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. AASP-32, No 2, pp. 262-271, April 1984.
- [Jin and Hauptmann, 2000a] R. Jin and A.G. Hauptmann, Cross Lingual Title Generation: Initial Steps, *Workshop on Interactive Searching in Foreign-Language Collections*, Human-Computer Interaction Laboratory, University of Maryland, College Park, MD. June 1, 2000. <http://www.clis.umd.edu/conferences/hcil00>
- [Jin and Hauptmann, 2000b] R. Jin and A.G. Hauptmann, Title Generation for Spoken Broadcast News using a Training Corpus, In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, P.R.China, 2000
- [Brown *et al.* 1990] P. Brown, S. Cocke, S. Della Pietra, Della Pietra, F. Jelinek, J. Lafferty, R. Mercer, and Roossin, A Statistical Approach to Machine Translation, *Computational Linguistics* V. 16, No. 2, June 1990.

Dealing with Dependencies between Content Planning and Surface Realisation in a Pipeline Generation Architecture

Kalina Bontcheva and Yorick Wilks

Department of Computer Science, University of Sheffield, 211 Portobello St., Sheffield S1 4DP, UK
{kalina,yorick}@dcs.shef.ac.uk

Abstract

The majority of existing language generation systems have a pipeline architecture which offers efficient sequential execution of modules, but does not allow decisions about text content to be revised in later stages. However, as exemplified in this paper, in some cases choosing appropriate content can depend on text length and formatting, which in a pipeline architecture are determined after content planning is completed. Unlike pipelines, interleaved and revision-based architectures can deal with such dependencies but tend to be more expensive computationally. Since our system needs to generate acceptable hypertext explanations reliably and quickly, the pipeline architecture was modified instead to allow additional content to be requested in later stages of the generation process if necessary.

1 Introduction

The astonishing explosion of the World Wide Web is leading to a growing need to personalise user experience in cyberspace (e.g., myYahoo, Amazon's book recommendations). Since research in Natural Language Generation (NLG) has already investigated ways to tailor automatically-generated texts to user goals and characteristics (e.g., [Paris, 1993; Zuckerman and McConachy, 1995]), these methods could be used to generate *dynamic hypertext*¹ which takes into account the interaction context and user preferences and characteristics.

Since users expect real-time interaction, efficient and robust applied NLG techniques are typically used for hypertext generation. For instance, ILEX [Knott *et al.*, 1996] uses a combination of canned stories and templates; EXEMPLARS [White, 1998] is rule-based; and PEBA [Milosavljevic *et al.*, 1996] uses text schemas [McKeown, 1986] and a phrasal lexicon. Also for efficiency reasons, dynamic hypertext generation systems have pipeline architectures where modules are executed sequentially and no module later in the architecture can request information from an earlier module. For example,

¹In *dynamic hypertext* page content and links are created on demand and are often adapted to the user and the previous interaction.

in such an architecture it is not possible to take into account text formatting and length (which are determined towards the end) when choosing the text content (which happens in the beginning).

The goal of our dynamic hypertext generation system, HYLITE+, is to produce encyclopaedia-style explanations of domain terminology (see Figure 3 below). The corpus analysis of online encyclopaedia [Bontcheva, 2001] and previous empirical studies (e.g., [Reinking and Schreiner, 1985]) have shown the positive effect of additional information – e.g., definition of key vocabulary, less-technical content, supply of background information and illustrations – on the subjects' reading comprehension and reading behaviour. On the other hand, hypertext usability studies [Nielsen, 2000] have shown that people read 25% slower on the screen, so hypertext needs to be concise with formatting, that facilitates skimming. Our empirical studies have shown [Bontcheva, 2001] that users prefer different additional information depending on the chosen formatting and desired explanation length.

This paper discusses several ways to provide additional information about unknown terms in generated encyclopedic entity descriptions. When such information is needed, the most appropriate clarification needs to be chosen depending on formatting, user knowledge and constraints (e.g., concise versus detailed pages). Each alternative requires different text content to be selected at the start of the generation process but the choice of alternative can only happen after the content and formatting for the main description have already been determined. Therefore, the original pipeline architecture was extended to allow limited module feedback. In the resulting *recursive pipeline architecture* additional content can be requested in later stages of the generation process (e.g., during surface realisation).

In effect, the content planner first produces the text plan for the concise hypertext which contains only facts about the explained concept (e.g., personal computer). Then during surface realisation, after formatting has been decided, the most suitable adaptivity alternative is chosen. Often this leads to the posting of a new communicative goal, which results in expanding the basic text with extra information.

The paper is structured as follows. Section 2 describes briefly HYLITE+ – the dynamic hypertext generation system in the context of which the *recursive pipeline* architecture (Section 3) was developed. Section 4 exemplifies the use

of the architecture for generating additional information that clarifies terms unknown to the user. The approach is also put in the context of previous work on interleaved, opportunistic, and revision-based language generation (Section 5). Finally the paper concludes with a discussion of some known limitations and future work.

2 HYLITE+ in a Nutshell

HYLITE+ generates encyclopedic explanations of terms in the chemical and computer domains. The user interacts with the system in an ordinary Web browser (e.g., Netscape, Internet Explorer) by specifying a term she wants to look up. The system generates a hypertext explanation where further information can be obtained by following links or specifying another query. Similar to all Web applications (see [Nielsen, 2000]), HYLITE+ needs to (i) respond in *real-time*, i.e., avoid algorithms with associated high computational cost; and (ii) be *robust*, i.e., always produce a response. Consequently the system uses some efficient and well-established applied NLG techniques such as text schemas and a phrasal lexicon (see [Reiter and Dale, 2000]).

Similar to other applied systems (see [Reiter, 1994]), HYLITE+ was initially implemented as a single-pass pipeline system, i.e., the generation modules were executed sequentially. The system input specifies the concept to be explained and the system parameters chosen by the user (e.g., concise versus detailed descriptions). The output is the generated hypertext explanation in HTML format, which is viewed by the user in a conventional browser (see Figure 3 below).

The system consists of several modules organised in two main stages: (i) content organisation, which includes *content selection*, *text organisation* and *semantic aggregation*; and (ii) *surface realisation* modules [Bontcheva, 2001]. As shown in Figure 1, the high-level modules use a discourse history and an agent model, ViewGen, [Ballim and Wilks, 1991] which contains both the system domain knowledge and user beliefs, stored in nested environments. The surface realisation modules use language-specific resources such as lexicons, morphology, and grammars.

The text planner uses high-level discourse patterns similar to text schemas [McKeown, 1986] which have been derived from analysing entries in encyclopedia and terminological dictionaries [Bontcheva, 2001]. For instance, entities (i.e., concepts inheriting from ENTITY in the hierarchy) are defined by their *supertype(s)* or *type definition*, *characteristics*, *functions*, *constituents* and *examples*. If the entity has several synonymous terms, the query one is used throughout the explanation and the rest are given in brackets when the entity is first introduced.

3 Adding Recursion to the Pipeline Architecture

One way of improving the user understanding of the generated hypertext is to clarify important unknown terms that occur in definitions, descriptions of parts and subtypes (for further detail see [Bontcheva, 2001]). However, different types of such information – definitions, familiar superconcepts, separate paragraphs, or just links – are preferred at dif-

ferent times, mainly depending on page length, formatting, and user preference for concise versus detailed explanations. In a sequentially organised pipeline architecture some of this information can be generated only if the necessary additional facts have already been extracted during content selection and incorporated in the text plan.

For instance, the decision whether it is appropriate to use a parenthetical definition or a paragraph-length description depends on their length, the chosen hypertext formatting, and the target length for the generated page². However formatting and length information are only available during surface realisation, not content planning, so HYLITE+ either has to extract in advance clarifying information for each unknown concept, or the architecture needs to be modified to allow such information to be requested later.

Selecting all potentially useful clarifying information in advance is likely to be computationally expensive as the text size grows. Even if such pre-selection is computationally feasible, the content planner still cannot commit to a particular strategy (e.g., a parenthetical definition or a familiar superconcept) because it needs to know, among other things, the length of the definition text itself³.

One way to resolve this problem is to implement a text revision mechanism which would analyse the content and structure of the generated hypertext and choose between alternative ways of including clarificatory information. However, despite the gains in fluency and coherence, revision-based approaches tend to suffer from computational problems due to the large number of alternatives that need to be explored (e.g., [Robin and McKeown, 1996]).

Also, since HYLITE+ was built on the basis of an existing pipeline-based generation system, our goal was to find a solution that would both provide the needed module feedback functionality and work in combination with the existing sequentially-designed modules. In other words, we needed a modification of the pipeline, that will allow the surface realiser to invoke the generator with new goals, so additional information is extracted and included only when appropriate.

Consequently, the existing pipeline architecture was modified to allow later stages to request additional information to be included (see Figure 1). A special monitoring module was added to detect opportunities for adapting the generated hypertext. The monitor uses the complete text plan, received from the content planning stage, together with the propositions realised to this point to estimate the length of the main explanation. Based on this information, user preferences, and the chosen document formatting, the monitor decides whether to post additional high-priority goals on the generator's agenda (e.g., `define(microprocessor)`: a goal to define a concept, `example(microprocessor)`: give an example).

When such goals are posted to the generator, surface realisation is temporarily suspended while the text for the new

²[Bontcheva, 2001] discusses the results of an empirical study of user acceptance of hypertext adaptivity techniques and the influence of length, formatting, and user characteristics on this choice.

³The user study showed that definitions longer than 10 words should not be provided in brackets because they disturb the text flow.

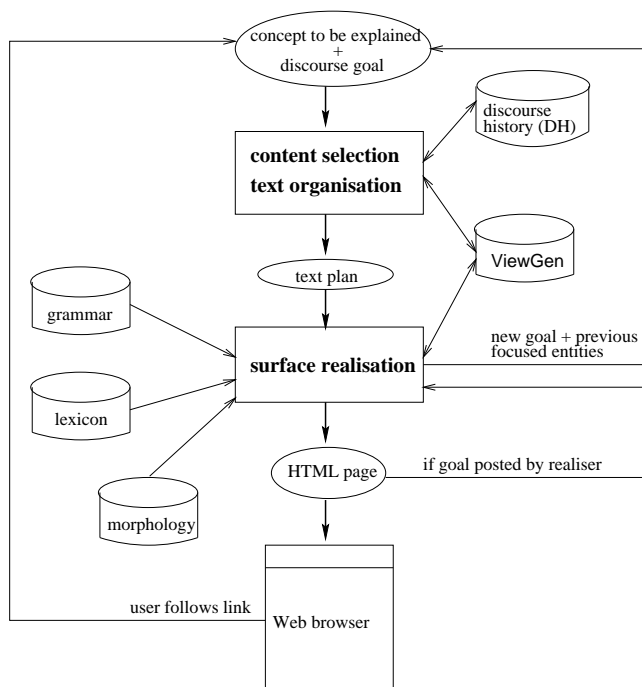


Figure 1: The HYLITE+ recursive pipeline architecture

goal is generated. This effectively opens a new discourse segment [Grosz and Sidner, 1986], which is dominated by the main explanation. All previously mentioned entities become part of the new focus space, so the generator can refer to them if necessary. The monitor also specifies some system parameters which influence the form of the newly generated text. For example, term definitions in brackets need to be noun phrases, instead of full sentences (see example in Figure 2), so the system configuration is changed from the default preference for sentences. When the text corresponding to the new goal is generated, the monitoring module evaluates the result and, if suitable, integrates it with the main text. Finally, text realisation is resumed until a new interruption occurs or the entire text plan has been verbalised.

If the newly generated text is found to be unsuitable (e.g., the definition is too long and will disturb the flow of the main text), the monitoring module tries another adaptivity technique if such is available (e.g., provide a known superconcept instead). Otherwise it leaves the text content as originally planned and the unknown concept is realised with its corresponding term with a hypertext link to a separate page.

In effect, the content planner first produces the text plan for the concise, unadapted hypertext which contains only facts about the explained concept (e.g., personal computer). Then during surface realisation, when formatting has been decided, the most suitable adaptivity alternative is chosen. In some cases this leads to the posting of a new communicative goal to the generator, which results in expanding the basic text with extra information.

Below we will show how the recursive pipeline is used for providing clarifying information about subtypes, parts, and unknown concepts in term definitions. The present experi-

ence shows that the efficiency of the pipeline architecture and the schemas is retained while the generator's scope and flexibility is improved.

4 Clarifying Unknown Terms

The content planner uses information from the user model to detect and annotate unknown concepts [Bontcheva and Wilks, 1999]. During surface realisation hypertext links, leading to separate explanation pages, are added for these concepts. In addition, some clarifying information is considered for unknown concepts in definitions, parts and subtypes.

The system mockup experiments [Bontcheva, 2001] showed that users prefer two types of concise parenthetical information: brief term definitions and a familiar superconcept. For longer texts, paragraph-long clarifying information can be provided in a separate section, containing more detail than just the definition/superconcept. Here we will only provide examples of using the recursive pipeline architecture to generate additional term definitions and paragraph-long explanations. Further details and a thorough discussion of the other adaptivity techniques are available in [Bontcheva, 2001].

4.1 Generating the definitions

Let us assume a user who has looked up computer programs and then followed a link to **personal computer**; the user has not specified preferences for types of clarifying information, so definitions can be provided where appropriate. Following this request, the content planner passes the following facts for realisation (the fact listing all parts is truncated here⁴):

```
[PC] <- (ISA) <- [MICRO_COMP fs(um_state:unknown)].
[PC] <- (PART_OF) <- [CPU fs(um_state:unknown)]
      <- (PART_OF) <- [MEMORY fs(um_state:unknown)]
      <- (PART_OF) <- [HDD fs(um_state:unknown)]
      <- (PART_OF) <- [DISPLAY]...
```

First the realiser determines the document formatting; in this case a bullet list is chosen to enumerate all parts. Then it starts generating text for the first graph. Because the introduced supertype is unknown, but important for the understanding of the text, the realisation monitor decides to provide some extra material about it. The action corresponding to generating a new definition is `define(micro_comp)`, which is passed as a parameter to the recursively called generator. The generation parameters are also set to very short texts with syntactic preference for noun phrases. The resulting text – "a small computer that uses a microprocessor as its central processing unit" – is evaluated for length and provided in brackets (see Figures 2 and 3).

Similarly for all unknown parts of the PC, the realisation monitor calls the generator recursively and obtains their definitions. Since the parts are described in a bullet list, it is more appropriate to provide their definitions with dashes, instead of brackets (see Figure 3).

⁴The facts are encoded as conceptual graphs, a type of semantic network, where concepts are written in square brackets and relations in round ones. Extra information (e.g., whether a concept is familiar to the user) can be associated with concepts and graphs as feature structures.

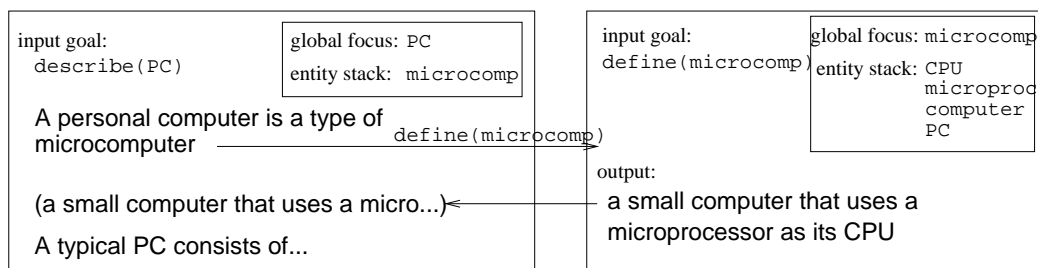


Figure 2: An example of clarifying information added during surface realisation

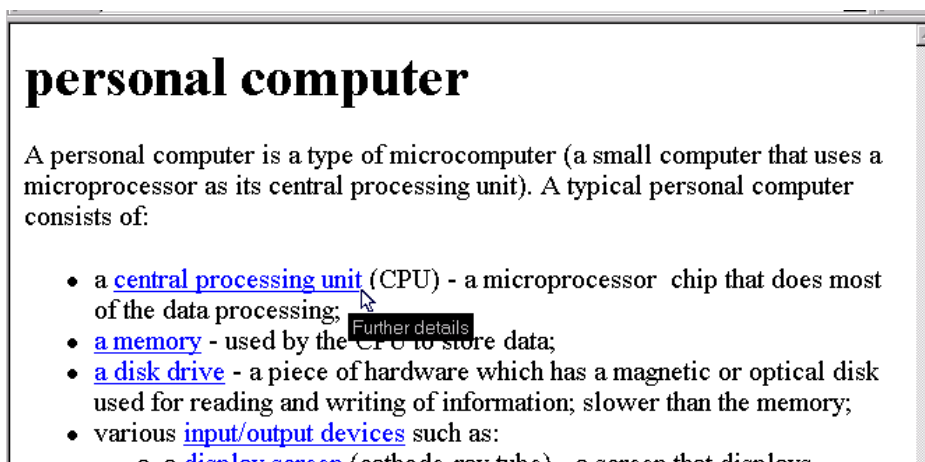


Figure 3: Example of clarifying definitions integrated in the main text

The realisation monitor has a set of rules which examine the type of proposition (e.g., definition, part_of, attribute) and the formatting of the main text, in order to determine the best formatting for the new material. At present, new material which is to be integrated in unformatted text is put in brackets. Since such parentheses disturb the flow of the main text, they are only provided once per sentence, usually for the most important concept in the proposition (e.g., the supertype). Other important unknown concepts in the same sentence are supplemented with a familiar supertype.

When a concept has been defined in brackets, the generated hypertext link also has a tag saying *Further information* to indicate that more information is available and can be reached by following the link (see Figure 3). If the knowledge base contains only the definition and no other information, then no link is provided because all relevant material has already been included in the current page.

4.2 Generating Paragraph-Length Clarifying Descriptions

So far we have discussed the generation of concise hypertext, where the main `describe(entity)` goal is realised in a schema-based, sequential fashion and brief additional definitions of unknown terms are included by posting additional high-priority goals. However, our empirical studies showed that in some cases users can prefer longer texts.

The corpus study showed that long encyclopedic articles

often provide detailed descriptions of subtypes and/or object parts. Therefore, one way to adapt longer generated hypertext is to provide paragraph-length descriptions of the unknown subtypes/parts, instead of including just their definitions or familiar supertypes. The experiments showed that such pages are preferred when users need more detailed information on the topic (e.g., for a school essay); in this case, the additional material is best organised in a separate section.

HYLITE+ generates such longer additional material in separate sections based on new communicative goals posted on its agenda by the monitor (see Figure 1). For example, if the main explanation contains propositions about several unknown object parts (e.g., computer parts) and more detailed texts are preferred by the user, a new `describe_all(part_of, computer)` goal is posted on the agenda. In this case, the goal is not marked as a high-priority one, so surface realisation is not interrupted. After the main explanation is completed, the generator fetches the new goal and starts a new section in the document describing all computer parts.

The `describe_all(Rel, X)` goal is decomposed in the following way:

```
forall C where Rel(C, X)
    describe(C)
```

i.e., describe all concepts *C* for which the relation *Rel* holds with *X*, e.g. describe all parts of the computer. Apart from the new goal, the generator is also passed parameters that specify

to generate the explanations as paragraphs, not whole pages.

In this way the generated document is regarded as an ordered set of goals and the text for each one of them is generated separately. This separation approach has the benefit of breaking down the text organisation problem into smaller parts which (i) can be more efficient to compute, but also (ii) can use different text organisation approaches for the different parts, e.g., schemas for the more ‘rigid’ parts and planning otherwise.

5 Related Work

The way in which the recursive pipeline architecture handles the dependencies between content planning and surface realisation bears some similarity with *interleaved language generation*. For example, [Rubinoff, 1992] describes an approach where the planner obtains feedback from the linguistic component in the form of a set of possible ways of expressing each plan element; the most suitable one is then added to the utterance. Some interleaved systems (e.g., [Finkler and Neumann, 1989; De Smedt, 1990]) are also *incremental*, i.e., their planner and realiser operate in parallel on different segments of an utterance. The most similar interleaved system is PAULINE [Hovy, 1988] where the planner produces only a partial text plan and makes the remaining commitments only when the realisation component needs them. In this way unexpected syntactic opportunities can be exploited and long-term goals (e.g., conciseness, politeness) taken into account during planning as well as realisation.

The main difference between the recursive pipeline approach and interleaved systems is that the latter tend to use feedback for every choice that depends on information from another module. In HYLITE+ the main text is always produced in a sequential fashion; interleaved planning and realisation only occur if the user preferences allow the use of adaptivity techniques that lead to new goals. Although potentially less flexible, the recursive approach enables the use of efficient techniques (schemas, sequential execution) to compute the main text reliably and quickly.

Another approach to the content adaptivity problem was explored in the ILEX system [Mellish *et al.*, 1998], which uses *opportunistic* planning to tailor the hypertext descriptions of museum exhibits. The planning mechanism is based on a structure called *text potential* – a graph of facts connected with thematic and rhetorical relations. Facts are chosen depending on their connection to the focus of the page (the described entity), interest and importance weights. This approach, however, seems difficult to apply to generated encyclopaedic-style explanations because it requires the creation of all links and weights in the text potential. Assigning values for importance and interest are particularly difficult because they both depend on the user goal (which is unknown to the system) and can change from one session to another (e.g., in-depth reading for a school essay versus looking up an unknown term).

The pipeline-based STOP system [Reiter, 2000] operates under very strict text size constraints which proved difficult to estimate at the content planning stage. In order to improve the generation results, Reiter experimented with a

multiple-solution pipeline⁵ (MSP) and revision-based variants of the system. The results showed that multiple-solution pipelines (for 4 solutions) and revision-based NLG are best suited for the task. The processing time per document is naturally higher – linearly increasing with the number of generated alternatives for the multiple-solution approach and the revision-based one is even slower than the MSP. The recursive pipeline architecture discussed here will not be suitable for such tightly constrained texts because the length of the yet unrealised propositions can only be approximated, which might lead to text overflow.

6 Conclusion and Future Work

This paper presented the *recursive pipeline architecture* developed as part of the dynamic hypertext generation system HYLITE+. The approach allows additional content to be requested in later stages of the generation process and integrated in the main explanation. The architecture also allows new goals to be executed after the main body of the explanation has been generated.

Although interleaved and revision-based approaches offer more flexibility, the advantages of the recursive pipeline is that it can be added to an already implemented sequential system with relatively minor modifications to the code. In addition, with our approach the main text is always computed in a sequential fashion, based on efficient applied NLG techniques (e.g., schemas, phrasal lexicon). Interleaved planning and realisation only occur if the user preferences allow the use of adaptivity techniques that lead to new goals.

In the case when the additional text is generated during surface realisation, the monitor determines the overall text length based on the length of the already realised facts plus an estimated length for all unrealised ones. Therefore the result is only an approximation of the final length, which is sufficient when, as in hypertext, size constraints are important but not very strict.

The results from a small-scale formative evaluation have confirmed that the majority of users prefer the adapted texts to the neutral version where no additional information is provided for the unknown terms. More exhaustive subject-based evaluation is currently being undertaken and the goal is to gain further insight into the users’ acceptance of hypertext adaptivity.

Performance-based evaluation of the recursive pipeline versus the baseline system (a version that does not provide additional information, just generates the main explanation) revealed that the new goals only add between 5% and 20% to the overall execution time – e.g., 5-10% for additional information which is less than 33% of the baseline explanation; 20% for additional information which is 50-60% of the baseline one. With both systems the overall processing time for a half page of hypertext is less than 1.5 seconds – a response time which allows users to stay focused on the interaction without any need for extra feedback [Nielsen, 2000].

⁵ Several different document plans are produced and after surface realisation a choice module selects the one that provides most content within the given size limit.

In the future we plan to experiment with using different content organisation strategies to generate different parts of the text. For example, following the ideas of [Paris, 1993] different kinds of knowledge can be provided depending on the user's level of expertise. Also, in some cases schemas can be replaced with text planning based on rhetorical relations, e.g., [Moore, 1995; Power, 2000]. In this way the more 'rigid' parts of the text can be generated efficiently with schemas, while more powerful but computationally expensive planning techniques are used only when necessary.

Acknowledgements

We wish to thank Hamish Cunningham and the anonymous reviewers for their helpful comments and suggestions.

References

- [Ballim and Wilks, 1991] A. Ballim and Y. Wilks. *Artificial Believers*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.
- [Bontcheva and Wilks, 1999] Kalina Bontcheva and Yorick Wilks. Generation of adaptive (hyper)text explanations with an agent model. In *Proceedings of the European Workshop on Natural Language Generation (EWNLG'99)*, Toulouse, France, May 1999.
- [Bontcheva, 2001] Kalina Bontcheva. *Generating Adaptive Hypertext Explanations with a Nested Agent Model*. PhD thesis, University of Sheffield, 2001. Forthcoming.
- [De Smedt, 1990] Koenraad De Smedt. IPF: An incremental parallel formulator. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 167–192. Academic Press, New York, 1990.
- [Finkler and Neumann, 1989] Wolfgang Finkler and Günter Neumann. POPEL-HOW: A distributed parallel model for incremental natural language production with feedback. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, volume 2, pages 1518–1523, Detroit, MI, August 20–25, 1989.
- [Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intentions and the structure of discourse. *Computational Linguistics Journal*, 12(3):175–204, 1986.
- [Hovy, 1988] Eduard H. Hovy. *Generating natural language under pragmatic constraints*. Lawrence Erlbaum, Hillsdale, New Jersey, 1988.
- [Knott et al., 1996] Alistair Knott, Chris Mellish, Jon Oberlander, and Mick O'Donnell. Sources of flexibility in dynamic hypertext generation. In *Proceedings of the 8th International Workshop on Natural Language Generation (INLG'96)*, 1996.
- [McKeown, 1986] Kathleen R McKeown. Discourse strategies for generating natural-language text. In B. L. Webber B. Grosz, K. S. Jones, editor, *Readings in Natural Language Processing*. Morgan Kaufmann Publishers, 1986.
- [Mellish et al., 1998] Chris Mellish, Mick O'Donnell, Jon Oberlander, and Alistair Knott. An architecture for opportunistic text generation. In *Proceedings of the International Natural Language Generation Workshop IWNLG'98*, 1998.
- [Milosavljevic et al., 1996] Maria Milosavljevic, Adrian Tulloch, and Robert Dale. Text Generation in a Dynamic Hypertext Environment. In *Proc. of 19th Australian Computer Science Conference*, Melbourne, 1996.
- [Moore, 1995] Johanna D. Moore. *Participating in Explanatory Dialogues*. MIT Press, Cambridge, MA, 1995.
- [Nielsen, 2000] Jakob Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, 2000.
- [Paris, 1993] Cécile L. Paris. *User modelling in text generation*. Francis Pinter Publishers, London, 1993.
- [Power, 2000] Richard Power. Planning by constraint satisfaction. In *Proceedings of COLING'2000*, 2000.
- [Reinking and Schreiner, 1985] David Reinking and Robert Schreiner. The effects of computer-mediated text on measures of reading comprehension and reading behaviour. *Reading Research Quarterly*, Fall:536–551, 1985.
- [Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, Cambridge, U.K., 2000.
- [Reiter, 1994] Ehud Reiter. Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In *Proceedings of 7th Int. Workshop on NL Generation (INLG-94)*, pages 163–170, Kennebunkport, Maine, USA, 1994.
- [Reiter, 2000] Ehud Reiter. Pipelines and size constraints. *Computational Linguistics*, 26:251–259, 2000.
- [Robin and McKeown, 1996] Jacques Robin and Kathy McKeown. Empirically designing and evaluating a new revision-based model for summary generation. *Artificial Intelligence*, 85(1-2), 1996.
- [Rubinoff, 1992] Robert Rubinoff. Integrating text planning and linguistic choice. In *Aspects of Automated Natural Language Generation*, Lecture Notes in Artificial Intelligence, 587, pages 45–56. Springer Verlag, Berlin, April 1992.
- [White, 1998] Michael White. Designing dynamic hypertext. In *2nd Workshop on Adaptive Hypertext and Hypermedia*, June 1998. Held in conjunction with Hypertext'98, Pittsburgh, USA.
- [Zuckerman and McConachy, 1995] Ingrid Zuckerman and Richard McConachy. Generating explanations across several user models: Maximizing belief while avoiding boredom and overload. In *Proceedings of 5th European Workshop on Natural Language Generation (EWNLG-95)*, 1995.

Narrative Prose Generation

Charles B. Callaway

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206 USA
cbcallaw@eos.ncsu.edu

James C. Lester

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206 USA
lester@csc.ncsu.edu

Abstract

Story generation is experiencing a revival, despite disappointing preliminary results from the preceding three decades. One of the principle reasons for previous inadequacies was the low level of writing quality, which resulted from the excessive focus of story grammars on plot design. Although these systems leveraged narrative theory via corpora analyses, they failed to thoroughly extend those analyses to all relevant linguistic levels. The end result was narratives that were recognizable as stories, but whose prose quality was unsatisfactory.

However, the blame for poor writing quality cannot be laid squarely at the feet of story grammars, as natural language generation has to-date not fielded systems capable of faithfully reproducing either the variety or complexity of naturally occurring stories. This paper presents the AUTHOR architecture for accomplishing precisely that task, the STORYBOOK implementation of a narrative prose generator, and a brief description of a formal evaluation of the stories it produces.

1 Introduction

Despite extensive research in the fields of story generation and natural language generation, collaborative research between the two has been virtually nonexistent. A major reason for this is the difficult nature of the problems encountered respectively in these fields. Story generators [Meehan, 1977; Yazdani, 1982; Lebowitz, 1985; Turner, 1994; Lang, 1997], typically address the macro-scale development of characters and plot, slowly refining from the topmost narrative goal level down to individual descriptions and character actions by progressively adding more and more detail. Meanwhile, work in natural language generation (NLG) focuses on linguistic phenomena at the individual sentence level, and only recently have NLG systems achieved the ability to produce multi-paragraph text. What remains is a substantial gap between the narrative plans produced by story generators and the requirements of NLG systems.

This is explained by the historic research programs of these two distinct fields. Story generation originally descends from

the application of planning formalisms to the work of sociolinguists such as Vladimir Propp [Propp, 1968], who created story grammars to capture the high-level plot elements found in Russian folktales. Early work (Figure 1) in this area [Meehan, 1977; Yazdani, 1982; Lebowitz, 1985] focuses on the creation of characters and their interactions with plot elements. In the latest of these, Lebowitz states, "Eventually, we expect UNIVERSE to be able to generate connected stories in natural language form over a long period of time. For the moment, we are concentrating on generating plot outlines, and leaving problems of dialogue and other low-level text generation for later." Moreover, even the most recent story generation systems, such as MINSTREL and JOSEPH [Turner, 1994; Lang, 1997], focus on characters and plot when generating text, without considering the actual linguistic structures found in the texts they are attempting to mimic (Figure 2).

However, the lack of progress in achieving computer-produced stories characterized by high-quality prose is far from one-sided. Rather than *narrative generation*, most full-scale NLG systems [Hovy, 1993; Young, 1996; Horacek, 1997; Lester and Porter, 1997; Mittal *et al.*, 1998; Callaway *et al.*, 1999] instead focus on *explanation generation*, creating scientific or instructional text which significantly differs in the distribution and frequency of syntactic, semantic, and orthographic features from that found in narrative prose (although a few projects do address some of these issues, *e.g.*, [Kantrowitz and Bates, 1992; Robin, 1994; Doran, 1998; Cassell *et al.*, 2000]). In addition, the most advanced of these systems are still not capable of producing more than two paragraphs of text, while the vast majority of naturally occurring narratives are at least several pages long. Finally, none of these systems are intended to accept narrative plans from a typical story generator.

To bridge the gap between story generators and NLG systems, we have developed the AUTHOR narrative prose generation architecture [Callaway, 2000] to create high-quality narrative prose comparable to, and in some cases identical to, that routinely produced by human authors. This architecture has been implemented in STORYBOOK, an end-to-end narrative prose generation system that utilizes narrative planning, sentence planning, a discourse history, lexical choice, revision, a full-scale lexicon, and the well-known FUF/SURGE [Elhadad, 1992] surface realizer to produce multi-page stories in the Little Red Riding Hood fairy tale domain.

ONCE UPON A TIME GEORGE ANT LIVED NEAR A PATCH OF GROUND. THERE WAS A NEST IN AN ASH TREE. WILMA BIRD LIVED IN THE NEST. THERE WAS SOME WATER IN A RIVER. WILMA KNEW THAT THE WATER WAS IN THE RIVER. GEORGE KNEW THAT THE WATER WAS IN THE RIVER. ONE DAY WILMA WAS VERY THIRSTY. WILMA WANTED TO GET NEAR SOME WATER. WILMA FLEW FROM HER NEST ACROSS A MEADOW THROUGH A VALLEY TO THE RIVER. WILMA DRANK THE WATER. WILMA WASN'T VERY THIRSTY ANY MORE.

Figure 1: Prose generated by TALE-SPIN, 1977

Narrative prose differs linguistically from text found in explanatory and instructional passages in a number of ways:

- The existence of character dialogue with the accompanying difficulties of orthographic markers [Doran, 1998; Callaway, 2001], speaker-hearer relationships, locutional relations and manner clauses, interjections, and changes in pronominalization patterns. For instance, the following would never be found in explanatory text: “Beware the wolves,” her mother said in a hushed voice.
- Since explanatory text lacks dramatic characters, there is little need to include personal pronouns, highly idiomatic text about personal needs, or intentional desires such as wanting, needing, or knowing.
- Without character dialogue, explanatory text is usually able to get by using only present verb tenses with an occasional reference to events in the past when discussing sequences of processes. However, dialogue and the complex interactions between characters opens up the need to perform at least simplistic temporal reasoning and realizations in complex present, future and past tenses.
- Because human authors employ widely differing styles in narrative (*e.g.*, Hemingway vs. Joyce) as opposed to explanatory or instructional text which tries to adhere to stricter conventions, a narrative prose generator should be capable of mimicking those different types of styles.
- Finally, a narrative prose generator must conform to common prose formatting conventions, such as knowing when to force paragraph breaks and being able to generate written stylistic effects like onomatopoeia, regional dialects, and emphasis (*e.g.*, “Ewww!” “B-b-but, it’s s-s-scary!” “Mom, you CAN’T do that!”)

STORYBOOK is capable of reproducing these phenomena, and doing so in both grammatically correct English and passable Spanish [Callaway *et al.*, 1999; Callaway, 2001].

Upon receiving a high-level story specification from a narrative planner, STORYBOOK (1) structures it into paragraph and sentence-sized chunks, (2) conducts a discourse history analysis to determine indefinite references and pronominalizations, (3) performs a lexical choice analysis to increase variety among concepts and event relations, (4) maps actors, props and events to semantic/syntactic roles in full linguistic deep structures, (5) revises paragraph-sized groups of

one day it happened that peasant quarreled with the wife. when this happened, peasant felt distress. in response, peasant took a walk in the woods. peasant found a pit when he looked under the bush. when this happened, peasant desired to punish wife. in response, peasant made it his goal that wife would be in the pit. peasant tricked wife. wife was in the pit. peasant lived alone.

Figure 2: Prose generated by JOSEPH, 1997

deep structures via aggregation and reordering to eliminate the short, choppy sentences characteristic of text produced by discourse planning systems, and (6) performs surface realization with integrated formatting to produce narrative prose similar to that found in stories written by human authors.

To evaluate the quality of the narratives that STORYBOOK produces, we created a simplified narrative planner capable of generating two Little Red Riding Hood stories expressed in the required high-level story specification. We then created five versions of STORYBOOK variously ablating the discourse history, lexical choice, and revision components to produce a total of 10 story versions which were then formally evaluated by a panel of judges. The results showed significant differences between the inclusion or ablation of individual architectural components.

2 Narrative Representation

While most researchers in story generation utilize planning mechanisms or story grammars, a growing literature on narratology [Propp, 1968; Segre, 1988; Bal, 1997] posits that narrative consists of the *fabula*, or sum total of knowledge and facts about a narrative world, and the *suzjet*, or the ordering and specifics about what the author presents and at which position(s) it occurs in the linear narrative. The AUTHOR architecture adopts this view and computationalizes it to describe the requirements of a narrative planner and a narrative prose generator: the narrative planner is responsible for creating both the *fabula* and *suzjet*, while the narrative prose generator is responsible for converting them into textually recognizable narratives.

A narrative world is also populated with a large number of scenes, characters, props, locations, events, and descriptions. The STORYBOOK implementation explicitly represents this knowledge, which forms the basis of the *fabula*. Initially, the *fabula* contains only ontological information, including the existence of broad concepts such as *forest*, *cottage*, and *person*, and concept relations like *next-to*, *mother-of*, and *moves-toward*. STORYBOOK assumes that a narrative planner is responsible for constructing the specific concept instances that populate a particular story, *e.g.*, Little Red Riding Hood lives in Cottage001, which is her house, while her grandmother (Grandmother001) lives in a different house, Cottage002.


```

;;; Fabula Operators
(NewNarrative Meehan-Narrative000 Narrator001)
(AddActor George-Ant003 George-Ant Ant Male "George Ant")
(AddActor Wilma-Bird004 Wilma-Bird Bird
  Female "Wilma Bird")
(AddLocation Patch005 Patch-Area)
(AddLocation Ground006 Ground-Earth-Area)
(AddLocation Nest007 Nest-For-Birds)
(AddLocation Ash-Tree008 Ash-Tree)
(AddProp Water009 Water)
(AddLocation River010 River)
(AddLocation Meadow011 Meadow)
(AddLocation Valley012 Valley)
(AddAlias Wilma013 Wilma Wilma-Bird004 "Wilma")
(AddAlias George014 George George-Ant003 "George")

;;; Narrative Stream Primitives
(narration-mode historical-fairy-tale mixed-dialogue
  simple-syntax ascii-format narrated english)
(narrator-mode narrator001 third-person disembodied)
(prop-relationship living-near george-ant003 patch005)
(refinement region-of patch005 ground006)
(specification living-near process-step-type
  once-upon-a-time)
(prop-property exist-being nest007)
(specification exist-being location-in ash-tree008)
(prop-relationship living-in wilma-bird004 nest007)
(prop-property exist-being water009)
(specification exist-being location-in river010)
(refinement quantifier-value water009 some)
(define-event being-in015 being-in water009 river010)
(actor-intent knowing wilma013 being-in015)
(specification knowing thought-binder that-binder)
(define-event being-in016 being-in water009 river010)
(actor-intent knowing george014 being-in016)
(specification knowing thought-binder that-binder)
(actor-property personal-condition wilma013 thirsty-state)
(specification personal-condition time one-day)
(refinement intensifier thirsty-state very)
(define-event getting-near017 getting-near none water009)
(actor-intent wanting wilma013 getting-near017)
(refinement quantifier-value water009 some)
(actor-action flying-from wilma013 nest007)
(refinement belonging-to nest007 wilma013)
(specification flying-from across-path meadow011)
(specification flying-from through-path valley012)
(specification flying-from destination river010)
(actor-action drinking-action wilma013 water009)
(actor-property personal-condition wilma013 thirsty-state)
(specification personal-condition duration any-more)
(specification personal-condition polarity negative)
(refinement intensifier thirsty-state very)

```

Figure 3: Fabula and Narrative Stream for generating Fig. 1

In addition, STORYBOOK assumes that the narrative planner is responsible for creating a stream of narrative events (the *suzjet*) that defines the linear ordering of events and descriptions as well as for *content determination*, the NLG term for deciding if particular narrative details or underlying facts are ever mentioned at all (e.g., events can be “too obvious” or perhaps meant to be inferred, as in mystery novels). Also, linearity can vary between different versions of a single story: a strict chronological ordering that states Little Red Riding Hood meets a wolf before travelling to grandmother’s house wouldn’t necessarily hold in the *in medias res* version.

In order to computationalize the fabula and narrative stream so they can serve as an interface between a narrative planner and a narrative prose generator, the AUTHOR architecture defines a set of *fabula operators* which can be used to construct the fabula from the original story ontology, and a set of *narrative stream primitives* (Figure 3), which define

the presentational order and content determination as well as information about what purpose that particular content serves at that point in the narrative.

A typical fabula operator relates a new *concept instance* (indicated by a unique number at the end of the name) to either some element in the story ontology or a previously created concept instance. A typical narrative stream primitive consists of a *narrative directive*, which describes the purpose for its inclusion by the narrative planner as well as the relationship between its arguments. The ordered arguments of a narrative directive are directly tied to either the original concepts in the story ontology or the derived concept instances created by the fabula operators. Furthermore, a partial order is imposed on the narrative stream forcing dependent elements to follow their modifiers (e.g. in the phrase “some water” from Figure 3, “water” is introduced in a narrative primitive before the “some” quantifier is introduced).

STORYBOOK currently defines six different fabula operators as well as 35 narrative stream primitives that serve three main functions: *delimitation* (narrator and scene changes, dialogue marking, and formatting directives), *foundation* (important clause-level events and descriptions, rhetorical, intentional, and perlocutionary relations loosely based on Speech Act Theory [Austin, 1962; Searle, 1969]), and *modification* (descriptive elaboration, comparison, manner, reason, time, etc.) These have been sufficient to encode three distinctly different multi-page Little Red Riding Hood fairy tales and to allow STORYBOOK’s narrative prose generator to create the narrative texts for each.

Finally, STORYBOOK assumes that the fabula and narrative stream operate in an *interlingual* environment, where the knowledge base encodes world knowledge in a language-neutral format [Callaway *et al.*, 1999; Callaway, 2001]. Thus given a single fabula and narrative stream, we should be able to produce fairy tales (or other forms of fictional narratives) in a variety of languages. A significant benefit of this approach is that such a narrative prose generator could also be used to improve the output of a machine translation system in a manner analogous to that of story generation. Regardless of how they are determined, the fabula and narrative stream are sent along with a set of stylistic parameters to the narrative prose generator as described in the following section.

3 The AUTHOR Architecture

To reproduce the complex phenomena that characterize human-generated stories, an effective narrative prose generator must be comprehensive in scope. It must address all of the requirements inherent in sentence planning, lexical choice, formatting, revising, and surface realization. AUTHOR therefore takes a standard “pipelined” approach with components for each of these processes. With the exception of discourse planning, which is here replaced by a narrative planner, STORYBOOK is the first NLG system to incorporate all of these modules into an end-to-end multi-page generation system.

Upon receiving the fabula and narrative stream from the narrative planner, STORYBOOK (Figure 4) first structures it into paragraph and sentence-sized chunks. It then conducts a discourse history analysis to determine pronominalizations

and identify seen/unseen concepts. Next, it performs a lexical choice analysis to increase variety. It then maps actors, props and events to semantic/syntactic roles in full linguistic deep structures. Next it revises paragraph-sized groups of sentences via aggregation and reordering to increase propositional density before finally performing surface realization to produce narrative prose similar to that found in stories written by human authors.

3.1 Narrative Organization

Because the narrative stream (Figure 3) is generated by the narrative planner as one long sequence, it must be segmented into groups of narrative stream primitives which reflect natural boundaries such as changes in speaker during dialogue and shifts in scene or topic. Because of the partial order imposed on the narrative stream, this is a relatively straightforward process. In addition, the discourse history and lexical choice modules operate by combing through the narrative stream and recording data in order to make decisions about altering the narrative stream. Because these three procedures involve a similar iterative analysis, they are performed by a single architectural module called the *narrative organizer* whose purpose is to take a flat, linear narrative stream and impose a hierarchical narrative structure onto it.

After the narrative stream primitives have been segmented, the discourse history module opportunistically replaces concept instances with the appropriate definite/indefinite forms and pronominalizations. These features are used to decide when to replace the lexicalizations of concepts and concept instances with the appropriate new linguistic deep structure information.¹ For example, a decision to make “wolf” or “butter” be indefinite when they are first mentioned in the discourse context may result in an indefinite article for the count noun (“a wolf”) or an indefinite determiner or determiner sequence for the mass noun (“some butter”). Knowing whether a concept instance has been seen or not requires computing and tracking several *occurrence* properties for every concept instance:

- *Frequency*: How often a concept instance has been used (lexicalized vs. pronominalized).
- *Last-Usage*: If its most recent use was lexicalized.
- *Recency*: How many distinct concept instances have been used since it was last seen (including gender).
- *Distance*: The number of scenes, paragraphs, or dialogue turns since it was last seen.

Similarly, pronominalization decisions are made to replace repetitive instances of concept instances with appropriate pronouns (e.g., “Grandmother” with the single feminine pronoun “she/her”). Because knowing when an instance is repetitive involves using the same occurrence properties, the lexical chooser similarly checks for excessive repetition of concept

¹ These “replacements” are more accurately “annotations” on the narrative stream, because future changes by the revision component may alter the circumstances that lead to a particular noun phrase being pronominalized. For instance, the revision component may swap the order of two sentences or change the position of a circumstantial clause from leading a sentence to following it.

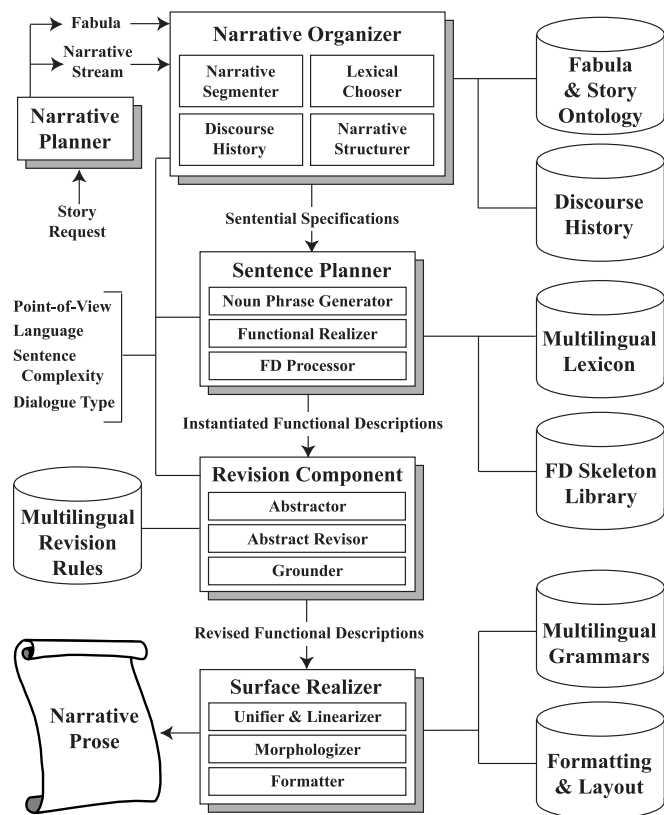


Figure 4: A Narrative Prose Generation Architecture

instances or relations. If this happens, the lexical chooser may replace elements of each narrative primitive with synonymous concept instances or relations from the fabula. The STORYBOOK lexical choice module detects:

- *Repetition in Noun Phrases*: Languages typically contain a large number of similar nouns. For example, Little Red Riding Hood might live in a house, cottage, shack, hut, cabin, etc.
- *Repetition in Verb Phrases*: Similarly, events have a number of lexicalizations. Little Red Riding Hood can walk through the forest, skip, amble, stroll, etc.
- *Repetition in Thematic Role Ordering*: Many event verbs also impose different theta frames even though they describe similar actions. Little Red Riding Hood might give her grandmother the cookies or grandmother might receive the cookies from Little Red Riding Hood.

Although this does not compare to more sophisticated methods [Elhadad, 1992; Stede, 1996] and is by no means suggested as a solution to the problem of lexical choice, it is sufficient to satisfy our goal of preventing repetitive prose. The result of segmentation, discourse history analysis, and lexical choice is thus a modified narrative stream. However, in classic pipelined NLG architectures, discourse planners typically produce a single structure (e.g., a frame) that corresponds to a sentence-sized chunk of the discourse plan. Thus, the job of the *narrative structurer* is to convert the groups of

narrative primitives into a sequence of specifications suitable for the sentence planner. Additionally, the narrative structurer is responsible for making decisions about tense shifting, especially for character dialogue. In dialogue, conversations usually take place in present tense even though the surrounding narrative is in past tense, and references to prior events are typically in the past tense where in expository text they would be in the past perfect tense (at least for English).

Because the fabula and story ontology exist by these stages, they can be used as knowledge sources for making appropriate decisions. For example, the discourse history module may examine the gender of a concept instance and thus know it should substitute “she” for “Little Red Riding Hood” without that knowledge having to be explicitly represented in the narrative stream. Similarly, the narrative structurer may examine the lexicon entry of a narrative stream primitive’s primary relation to determine its theta frame and its argument’s semantic type restrictions for error-checking purposes.

3.2 Sentence Planning

The function of the sentence planner is to take a specification for the semantic content of a sentence (or *protosentence*) and to plan the roles (either semantic or syntactic) that each of its elements play in a particular sentence. Because our approach utilizes an off-the-shelf surface realizer that expects particular semantic roles, we require that our sentence planner produce the deep structure linguistic representations known as *functional descriptions* (FDs, Figure 5). Functional descriptions are hybrid semantic/syntactic entities that can be used to produce text via unification with the FUF/SURGE [Elhadad, 1992] surface realizer.

A sentence planner must:

- Guarantee that complex content units are properly and completely packaged within functional descriptions, *e.g.*, complex noun phrases such as “the beautiful cottage where they lived” must be (a) capable of being created as a linguistic deep structure and (b) encapsulated so that it can be manipulated as a whole by succeeding elements of the pipelined NLG architecture.
- Assign thematic roles to concepts. To achieve semantic equivalence between a sentence’s frame specification and the corresponding deep structure, a sentence planner must ensure that relations in the specification are precisely mapped to the appropriate thematic roles in a functional description, *e.g.*, *mother001* could be mapped to *agent* and *nest007* to *located*.
- Robustly construct functional descriptions. A sentence planner must ensure that only FDs that will create grammatical sentences can be constructed. A number of errors that degrade robustness must be curbed, *e.g.*, lack of appropriate lexicalizations, missing semantic roles, and sentential modifiers that conflict with the overall sentential semantics.

Once the sequence of narrative stream primitives has been processed by the sentence planner, the resulting FDs (representing the deep linguistic structures for each protosentence) can be given directly to the surface realizer for text generation. However, because the quality of simple propositional

```
((cat clause)
 (tense past)
 (process ((type existential)))
 (participants ((located ((cat common)
                          (definite no)
                          (lex "nest")))))
 (pred-modif ((location ((cat pp)
                        (prep ((lex "in"))
                               (np ((cat common)
                                      (definite no)
                                      (lex "ash tree")))))))))
```

Figure 5: Functional Description (FD) for “There was a nest in an ash tree.” from Figure 1, Sentence 2.

sentences is notoriously poor, STORYBOOK revises them, iteratively saving each FD while maintaining the paragraph separations imposed by the narrative segmenter and proceeds to send paragraph-sized batches to the revision component (described in the following section) in order to improve overall prose quality.

3.3 Revision

Revision modules [Dalianis and Hovy, 1993; Robin, 1994; Callaway and Lester, 1997; Shaw, 1998] take a series of protosentences (simple sentences with limited content, *e.g.*, “The wolf saw Little Red Riding Hood”) and rearrange them by *aggregation*, *i.e.* combining protosentences in various ways, or by *migration*, *i.e.* permuting the order of two adjacent protosentences. The REVISOR component [Callaway and Lester, 1997] receives a paragraph-sized group of protosentences from the sentence planner represented as an ordered set of deep-structure functional descriptions.

To illustrate, consider the issue of clause aggregation, a central problem in multi-sentential text generation. Suppose a narrative prose generation system is given the task of constructing a fairy tale and it produces several pages of prose. Although it might accurately communicate the content of the narrative plan, the terseness of each sentence makes the overall effect disjointed; in other words, content without literary form. An entire story comprised solely of protosentences is intolerable for almost any adult reader. (See sample prose in Figures 1 and 2.)

To avoid producing a series of abrupt sentences, a narrative planner could be assigned the task of predicting how particular concepts will be realized in order to optimize clause aggregation and reordering decisions. However, this approach violates modularity considerations and does not scale well: it significantly complicates the design of the narrative planner by forcing it to attend simultaneously to content selection, narrative organization, and complex syntactic issues. Alternatively, the propositions could be grouped by a single-pass realization system. This approach is quite inefficient and also ineffective. Reorganizing, aggregating, and realizing the specifications in a single pass poses innumerable difficulties: the realizer would somehow have to anticipate the cumulative effects of all aggregation decisions with regard to grammaticality, subordination, and lexical choice.

An important aspect of revision in NLG is the concept of *discourse constraints*, which specify a partial order on the sequence of functional descriptions. For example, the narra-

tive planner might hand down a narrative constraint stating that, in a particular narrative passage, a sequence of events are causal in nature and that to reorder them in some fashion could destroy that causality in the mind of the reader. Additionally, because narrative prose includes character dialogue, it is important to prevent the reordering of character utterances. Thus, discourse constraints are employed to restrict aggregation and migration revisions that would affect particular types of clause elements across critical semantic boundaries. STORYBOOK utilizes the multilingual version of the REVISOR component described in [Callaway and Lester, 1997] to perform all of these tasks.

3.4 Surface Realization

The revision component passes the series of revised functional descriptions one by one to the surface realizer, which is responsible for producing the actual readable text that readers see. STORYBOOK employs the FUF surface realizer, which is accompanied by the SURGE (Systemic Reusable Grammar of English) grammar. SURGE, written as a *systemic grammar* [Halliday, 1976] in the FUF formalism, is the largest generation grammar in existence in terms of coverage, containing large portions of Quirk's Comprehensive Grammar of English [Quirk *et al.*, 1985] in an HPSG [Pollard and Sag, 1994] interpretation.

Modifications were made to SURGE to allow for dialogue orthography [Callaway, 2001], integrated formatting to produce LATEX, HTML, and XML, as well as a number of grammatical additions to account for syntactic constructions encountered during our corpus analyses (*i.e.*, linguistic phenomena we encountered in narratives that were not present in our analyses of explanatory and instructional text). This allows STORYBOOK to produce webpages as output that include pre-generated graphics specified in the narrative stream as well as boldface, italics, and font size embedded into individual sentences. Furthermore, we implemented a Spanish version of SURGE as described in [Callaway *et al.*, 1999] and also augmented it to produce character dialogue, *etc.*

4 Implementation and Evaluation

STORYBOOK is an end-to-end generation system capable of producing multi-page narrative prose in the Little Red Riding Hood domain like that found in Figure 6, which required 74 fabula operators and 253 narrative stream primitives to generate. STORYBOOK is implemented in HARLEQUIN LISP on a Dell Precision Workstation 410 using a 600 MHz Pentium III processor with 512 MB of memory. The initial story ontology consists of approximately 500 concepts and 300 relations (including their lexicon entries) covering three different Little Red Riding Hood narratives.

STORYBOOK consists of approximately 10,000 lines of Lisp (for narrative organization, sentence planning, and revision, but not surface realization). In addition, there are approximately 30 revision rules which are presently being modified to work for Spanish. The Spanish version of SURGE is approximately the same size as the English version. During story writing, surface realization is by far the largest consumer of time, usually requiring 90% of the 45–90 seconds needed to generate a two to three page narrative.

Once upon a time, a woodcutter and his wife lived in a small cottage. The woodcutter and his wife had a young daughter, whom everyone called Little Red Riding Hood. She was a merry little maid, and all day long she went singing about the house. Her mother loved her very much.

One day her mother said, "My child, go to grandmother's house. We have not heard from her for some time. Take these cakes, but do not stay too long. And, beware the dangers in the forest."

Little Red Riding Hood was delighted because she was very fond of her grandmother. Her mother gave her a well-filled basket and kissed her goodbye.

The road to grandmother's house led through the dark forest, but Little Red Riding Hood was not afraid and she went on as happy as a lark. The birds sang her their sweetest songs while the squirrels ran up and down the tall trees. Now and then, a rabbit would cross her path.

Little Red Riding Hood had not gone far when she met a wolf.

"Hello," greeted the wolf, who was a cunning-looking creature. "Where are you going?"

"I am going to my grandmother's house," Little Red Riding Hood replied.

"Ah, well then, take care in the forest, for there are many dangers." And then the wolf left.

Little Red Riding Hood was not in a hurry. Indeed, she gathered wild flowers and chased the pretty butterflies.

Meanwhile the wolf ran ahead very quickly and soon arrived at grandmother's house. He knocked on the door gently. The old lady asked, "Who is there?"

The wolf replied, "It is Little Red Riding Hood, your granddaughter."

And so the old lady opened the cottage door. The wolf rushed in immediately and devoured the lady in one bite. Then he shut the door and climbed into the old lady's bed.

Much later Little Red Riding Hood arrived at grandmother's house. She knocked on the door and shouted, "Grandmother, it is Little Red Riding Hood."

"Pull the string. The door will open."

And so Little Red Riding Hood opened the door and walked in. "Grandmother, what big eyes you have."

"All the better to see with, dear."

"Grandmother, what big ears you have."

"All the better to hear with, dear."

"And, grandmother, what big teeth you have!"

"All the better to eat up with!" yelled the wolf.

And then the wolf jumped up and devoured Little Red Riding Hood in one bite.

Figure 6: Example text produced by STORYBOOK

Previous story generation projects such as TALE-SPIN [Meehan, 1977], UNIVERSE [Lebowitz, 1985], and JOSEPH [Lang, 1997], which actually generated narrative prose were never subjected to an empirical evaluation to determine qualitatively or quantitatively how well their systems produced narratives. Also, narrative prose generation is currently at such an early stage of development that its evaluation should be conducted in a manner that is qualitatively different from work in more mature areas such as machine learning.

In order to assess the utility and overall contributions of our deep generation architecture to the task of narrative prose generation, we conducted a formal evaluation of the STORYBOOK system. Its purpose was to establish a baseline for future NLG systems by judging the performance of three key architectural components that differentiate shallow NLG systems from deep NLG systems: the discourse history module, lexical choice module, and revision module.

Formally comparing human-authored narratives with those produced by computer presents a difficult problem: there is no known objective metric for quantitatively evaluating narrative prose in terms of how it performs *as a story*. Simple metrics exist for evaluation at the sentence level (*e.g.*, number of words, depth of embedding, *etc.*), but a narrative *per se* cannot be considered to be merely a collection of sentences that are not related to each other. We instead opted for a computer vs. computer style of evaluation involving the ablation of the three architectural components mentioned above.

To stand in for the narrative planner (which is beyond the scope of this work), we created a modestly sized finite state automaton (containing approximately 200 states) capable of producing two stories, comprising two and three pages respectively. Furthermore, we fixed the content of those stories (*i.e.*, the fabula and narrative stream were identical) and ran five different versions of STORYBOOK on each story: (1) all three architectural components working, (2) revision turned off, (3) lexical choice turned off, (4) the discourse history turned off, and finally (5) a version with all three components turned off. This resulted in ten total narratives which we presented to our test subjects. While the two versions differed in the sense that particular modules were either ablated or not, the two stories differed because they were created from two separate paths through the planning automaton. Thus, Story #1 had some different events, descriptions, and props than Story #2 did.

Twenty test subjects graded each narrative over nine grading factors (representing various stylistic and linguistic criteria) according to an A–F scale. We then converted the results to a quantified scale where A = 4.0, B = 3.0, C = 2.0, D = 1.0, and F = 0.0 and tallied and averaged the final scores. To determine the quantitative significance of the results, we performed an ANOVA test over both stories. The analysis was conducted for three independent variables (test subject, story, and version) over the following grading factors:

- *Overall*: How is the story as an archetypal fairy tale?
- *Style*: Did the author use an appropriate writing style?
- *Grammaticality*: How would you rate the syntactic quality?
- *Flow*: Did the sentences flow from one to the next?

- *Diction*: How appropriate were the author's word choices?
- *Readability*: How hard was it to read the prose?
- *Logicity*: Did the story seem out of order?
- *Detail*: Did the story have the right amount of detail, or too much or too little?
- *Believability*: Did the story's characters behave as you would expect?

The results of the ANOVA analysis point to three significant classes of narratives due to the architectural design of the narrative prose generator. The most preferred narrative class, consisting of Versions (1) and (3), were not significantly different from each other while they were rated significantly higher than all other versions. In addition, Version (2) scored significantly better than the third class formed by Versions (4) and (5), each of which lacked a discourse history.

The results indicate that discourse history and revision components are extremely important, while lexical choice improved text significantly in Story #1 but not in Story #2. Upon analysis of the comments in their evaluations, it became clear that a principal reason was the test subjects' belief that the increased lexical variation might prove too difficult for children to read (even though we provided no indication that the target audience was children) and thus Version (1) compared less favorably to Version (3) due to the more complex and varied words it contained. It is not clear whether a lexical choice component would play a much more significant role in subject matter where a more adult audience was expected or if a larger-scale component were utilized.

5 Conclusions

Full-scale linguistic approaches to narrative prose generation can bring about significant improvements in the quality of text produced by story generators. By integrating off-the-shelf NLG components and adding a well-defined computational model of narrative, we can create a new generation of story systems whose written prose quality far surpasses that of previous attempts. This approach has been implemented in STORYBOOK, a narrative prose generator that produces multi-page fairy tales in near realtime. This deep structure approach has also been formally evaluated, suggesting that the architectural modules responsible for a significant improvement in prose quality are components not found in shallow (template-based) generation systems currently employed by most story generators. It is hoped that this work begins to bridge the traditional gap between story generators and NLG systems.

6 Acknowledgements

The authors wish to thank Joy Smith of NC State University for her help with the statistical analysis; Bruce Porter and Eric Eilerts for their development of the KM knowledge base language; Michael Elhadad and Jacques Robin for their development of FUF/SURGE; Michael Young for helpful discussion on narratives; the anonymous reviewers for their insightful comments. Support for this work was provided by the IntelliMedia Initiative of North Carolina State University.

References

- [Austin, 1962] J. L. Austin. *How to Do Things with Words*. Oxford University Press, New York, 1962.
- [Bal, 1997] Mieke Bal. *Narratology: Introduction to the Theory of Narrative, 2nd Edition*. University of Toronto Press, Toronto, Canada, 1997.
- [Callaway and Lester, 1997] Charles B. Callaway and James C. Lester. Dynamically improving explanations: A revision-based approach to explanation generation. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 952–58, Nagoya, Japan, 1997.
- [Callaway et al., 1999] C. Callaway, B. Daniel, and J. Lester. Multilingual natural language generation for 3D learning environments. In *Proceedings of the 1999 Argentine Symposium on Artificial Intelligence*, pages 177–190, Buenos Aires, Argentina, 1999.
- [Callaway, 2000] Charles Callaway. *Narrative Prose Generation*. PhD thesis, North Carolina State University, Raleigh, NC, 2000.
- [Callaway, 2001] Charles Callaway. A computational feature analysis for multilingual character-to-character dialogue. In *Proceedings of the Second International Conference on Intelligent Text Processing and Computational Linguistics*, pages 251–264, Mexico City, Mexico, 2001.
- [Cassell et al., 2000] J. Cassell, M. Stone, and H. Yan. Coordination and context-dependence in the generation of embodied conversation. In *International Natural Language Generation Conference*, Mitzpe Ramon, Israel, 2000.
- [Dalianis and Hovy, 1993] Hercules Dalianis and Eduard Hovy. Aggregation in natural language generation. In *Proceedings of the Fourth European Workshop on Natural Language Generation*, Pisa, Italy, 1993.
- [Doran, 1998] Christine Doran. *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1998.
- [Elhadad, 1992] Michael Elhadad. *Using Argumentation to Control Lexical Choice: A Functional Unification Implementation*. PhD thesis, Columbia University, 1992.
- [Halliday, 1976] Michael Halliday. *System and Function in Language*. Oxford University Press, Oxford, 1976.
- [Horacek, 1997] Helmut Horacek. A model for adapting explanations to the user's likely inferences. *User Modeling and User-Adapted Interaction*, 7(1):1–55, 1997.
- [Hovy, 1993] Eduard H. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385, 1993.
- [Kantrowitz and Bates, 1992] M. Kantrowitz and J. Bates. Integrated natural language generation systems. In R. Dale, E. Hovy, D. Rosner, and O. Stock, editors, *Aspects of Automated Natural Language Generation*, pages 247–262. Springer-Verlag, Berlin, 1992.
- [Lang, 1997] R. Raymond Lang. *A Formal Model for Simple Narratives*. PhD thesis, Tulane University, New Orleans, LA, 1997.
- [Lebowitz, 1985] M. Lebowitz. Story-telling as planning and learning. *Poetics*, 14(3):483–502, 1985.
- [Lester and Porter, 1997] James C. Lester and Bruce W. Porter. Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. *Computational Linguistics*, 23(1):65–101, 1997.
- [Meehan, 1977] J. Meehan. Tale-Spin, an interactive program that writes stories. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1977.
- [Mittal et al., 1998] V. Mittal, J. Moore, G. Carenini, and S. Roth. Describing complex charts in natural language: A caption generation system. *Computational Linguistics*, 24(3):431–469, 1998.
- [Pollard and Sag, 1994] C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, 1994.
- [Propp, 1968] V. Propp. *Morphology of the Folktale*. University of Texas Press, Austin, TX, 1968.
- [Quirk et al., 1985] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A Comprehensive Grammar of the English Language*. Longman Publishers, 1985.
- [Robin, 1994] Jacques Robin. *Revision-Based Generation of Natural Language Summaries Providing Historical Background*. PhD thesis, Columbia University, December 1994.
- [Searle, 1969] J. Searle. *Speech Acts*. Cambridge University Press, Cambridge, England, 1969.
- [Segre, 1988] Cesare Segre. *Introduction to the Analysis of the Literary Text*. Indiana University Press, Bloomington, IN, 1988.
- [Shaw, 1998] James Shaw. Segregatory coordination and ellipsis in text generation. In *COLING-ACL-98: Proceedings of the Joint 36th Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, pages 1220–1226, Montréal, Canada, 1998.
- [Stede, 1996] Manfred Stede. *Lexical Semantics and Knowledge Representation in Multilingual Sentence Generation*. PhD thesis, University of Toronto, Toronto, Ontario, 1996.
- [Turner, 1994] Scott R. Turner. *The Creative Process: A Computer Model of Storytelling and Creativity*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.
- [Yazdani, 1982] Masoud Yazdani. How to write a story. In *Proceedings of the European Conference on Artificial Intelligence*, Orsay, France, July 1982.
- [Young, 1996] R. Michael Young. Using plan reasoning in the generation of plan descriptions. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1075–1080, 1996.

NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

NATURAL LANGUAGE —
LEARNING FOR INFORMATION EXTRACTION

Adaptive Information Extraction from Text by Rule Induction and Generalisation

Fabio Ciravegna

Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street,
S1 4DP Sheffield, UK
F.Ciravegna@dcs.shef.ac.uk

Abstract

(LP)² is a covering algorithm for adaptive Information Extraction from text (IE). It induces symbolic rules that insert SGML tags into texts by learning from examples found in a user-defined tagged corpus. Training is performed in two steps: initially a set of tagging rules is learned; then additional rules are induced to correct mistakes and imprecision in tagging. Induction is performed by bottom-up generalization of examples in the training corpus. Shallow knowledge about Natural Language Processing (NLP) is used in the generalization process. The algorithm has a considerable success story. From a scientific point of view, experiments report excellent results with respect to the current state of the art on two publicly available corpora. From an application point of view, a successful industrial IE tool has been based on (LP)². Real world applications have been developed and licenses have been released to external companies for building other applications. This paper presents (LP)², experimental results and applications, and discusses the role of shallow NLP in rule induction.

1. Introduction

By general agreement the main barriers to wide use and commercialization of Information Extraction from text (IE) are the difficulties in adapting systems to new applications. The classical IE has been focusing on applications to free texts; therefore systems often rely on approaches based on Natural Language Processing (NLP) (e.g. using parsing) [Humphreys *et al.* 1997; Grishman 1997]. Most systems require the manual development of resources (e.g. grammars) by a user skilled in NLP [Ciravegna 2000]. There is an increasing interest in applying machine learning (ML) to IE in order to build adaptive systems. Up to now, the use of ML has been approached mainly in an NLP-oriented perspective, i.e. in order to reduce the amount of work to be done by the NLP experts in porting systems across free text based

scenarios [Cardie 1997; Miller *et al.* 1998; Yangarber *et al.* 2000]. Given the current technology, IE experts are still necessary.

In the last years, the increasing importance of the Internet has stressed the central role of texts such as emails, Usenet posts and Web pages. In this context, extralinguistic structures (e.g. HTML tags, document formatting, and ungrammatical stereotypical language) are elements used to convey information. Linguistically intensive approaches are difficult or unnecessary in such cases. For this reason a new research stream on adaptive IE has arisen at the convergence of NLP, Information Integration and Machine Learning. The goal is to produce IE algorithms and systems adaptable to new Internet-related applications/scenarios by using only an analyst's knowledge (i.e. knowledge on the domain/scenario itself) [Kushmerick 1997; Califf 1998; Muslea *et al.* 1998; Freitag and McCallum 1999; Soderland 1999; Freitag and Kushmerick 2000]. Such algorithms are very effective when applied on highly structured HTML pages, but less effective on unstructured texts (e.g. free texts). In our opinion this is because most successful algorithms make scarce (or no) use of NLP, tending to avoid any generalization over the flat word sequence. When they are applied to unstructured texts, data sparseness becomes a problem.

This paper presents (LP)², an adaptive IE algorithm designed in this new stream of research that makes use of shallow NLP in order to overcome data sparseness when confronted with NL texts, while keeping effectiveness on highly structured texts. This paper first introduces the algorithm, discusses experimental results and shows how the algorithm compares successfully with the current state of the art. The role and importance of shallow NLP for overcoming data sparseness is then discussed. Finally a successful industrial system for adaptive IE built around (LP)² is presented and some conclusions and future work are drawn.

2. The Rule Induction Algorithm

(LP)² learns from a training corpus where a user has highlighted the information to be extracted with differ-

ent SGML tags. It induces symbolic rules that insert SGML tags into texts in two steps:

1. Sets of **tagging rules** are induced by bottom-up generalization of tag instances found in the training corpus. Shallow knowledge about NLP is used in the generalization process.
 2. **Correction rules** are induced that refine the tagging by correcting mistakes and imprecision
- This section presents and discusses these two steps.

2.1 Inducing Tagging Rules

A tagging rule is composed of a left hand side, containing a pattern of conditions on a connected sequence of words, and a right hand side that is an action inserting an SGML tag in the texts. Each rule inserts a single SGML tag, e.g. *<speaker>*. This makes (LP)² different from many adaptive IE algorithms, whose rules recognize whole slot fillers (i.e. insert both *<speaker>* and *</speaker>* [Califf 1998; Freitag 1998]) or even multi slots [Soderland 1999]. The tagging rule induction algorithm uses positive examples from the training corpus for learning rules. Positive examples are the SGML tags inserted by the user. All the rest of the corpus is considered a pool of negative examples. For each positive example the algorithm: (1) builds an initial rule, (2) generalizes the rule and (3) keeps the *k* best generalizations of the initial rule. In particular (LP)²'s main loop starts by selecting a tag in the training corpus and extracting from the text a window of *w* words to the left and *w* words to the right. Each information stored in the 2**w* word window is transformed into a condition in the initial rule pattern, e.g. if the third word in the window is "seminar", the condition on the third word in the pattern will be word="seminar". Each initial rule is then generalized. In the generalization process (LP)² uses generic shallow knowledge about Natural Language as provided by a morphological analyzer, a POS tagger and a user-defined dictionary (or a gazetteer). A *lexical item* (LexIt in the following) summarizes such knowledge for each word (e.g., companies) via: a lemma (company), a lexical category (noun), case information (lowercase) and a list of user defined classes as defined by a user-defined dictionary or a gazetteer (if available). An initial rule and associated information in the LexIts is in Table 1.

Generalization consists in the production of a set of rules derived by relaxing constraints in the initial rule pattern. Conditions are relaxed both by reducing the pattern in length and by substituting constraints on words with constraints on some parts of the additional knowledge. Table 2 shows one of the many generalizations for rule in table 1. The last step of the algorithm is the selection of the best generalizations. Each generalization is tested on the training corpus and an accuracy score $L = \text{wrong}/\text{matched}$ is calculated. For each initial instance the *k* best generalizations are kept that: (1) report better accuracy; (2) cover more positive examples;

- (3) cover different parts of input¹; (4) have an error rate that is less than a specified threshold.

word index	Condition	Associated information				Action
	word	lemma	LexCat	case	SemCat	Tag
1	the	the	Art	low		
2	seminar	seminar	Noun	low		
3	at	at	Prep	low		<stime>
4	4	4	Digit	low		
5	pm	pm	Other	low	timeid	
6	will	will	Verb	low		

Table 1: Starting rule (with associated NLP knowledge) inserting *<stime>* in the sentence "the seminar at *<stime>* 4 pm will...".

The other generalizations are discarded. Retained rules become part of the **best rules pool**. When a rule enters the best rules pool, all the instances covered by the rule are removed from the positive examples pool, i.e. covered instances will no longer be used for rule induction ((LP)² is a covering algorithm). Rule induction continues by selecting new instances and learning rules until the pool of positive examples is void.

Word index	Condition					Action
	Word	Lemma	LexCat	Case	SemCat	Tag
3		at				<time>
4			digit			
5					timeid	

Table 2: A generalization for rule in table 1. The pattern is relaxed in length (conditions on words 1, 2 and 6 were removed) and conditions on the other words were substituted by other constraints.

2.1.1 Learning Contextual Rules

When applied on the test corpus, the best rules pool provides good results in terms of precision, but limited effectiveness in terms of recall. This means that such rules insert few tags (low recall), and that such tags are generally correct (high precision). Intuitively this is because the absolute reliability required for rule selection is strict, thus only some of the induced rules will match it. In order to reach acceptable effectiveness, it is necessary to identify additional rules able to raise recall without affecting precision. (LP)² recovers some of the rules not selected as best rules and tries to constraint their application to make them reliable. Constraints on rule application are derived by exploiting interdependencies among tags. As mentioned, (LP)² learns rules for inserting tags (e.g., *<speaker>*) independently from other tags (e.g., *</speaker>*). But tags are not independent. There are two ways in which they can influence each other: (1) tags represent slots, therefore *<tag_x>* always requires *</tag_x>*; (2) slots can be concatenated into linguistic patterns and therefore the presence of a slot can be a good indicator of the presence of another, e.g.

¹ Rules derived from the same seed cover the same portions of input when ineffective constraints are present.

`</speaker>` can be used as “anchor tag” for inserting `<stime>`². In general it is possible to use `<tagx>` to introduce `<tagy>`. (LP)² is not able to use such contextual information, as it induces single tag rules. The context is reintroduced in (LP)² as an external constraint used to improve the reliability of unreliable rules. In particular, (LP)² reconsiders low precision non-best rules for application in the context of tags inserted by the best rules only. For example some rules will be used only to close slots when the best rules were able to open it, but not close it (i.e., when the best rules are able to insert `<tagx>` but not `</tagx>`). Selected rules are called **contextual rules**. As example consider a rule inserting a `</speaker>` tag between a capitalized word and a lowercase word. This is not a best rule as it reports high recall/low precision on the corpus, but it is reliable if used only to close an open `<speaker>`. Thus it will only be applied when the best rules have already recognized an open `<speaker>`, but not the corresponding `</speaker>`. Area of application is the part of the text following a `<speaker>` and within a distance minor or equal to the maximum length allowed for the present slot³. “Anchor tags” used as contexts can be found either to the right of the rule space application (as in the case above when the anchor tag is `<speaker>`), or to the left as in the opposite case (anchor tag is `</speaker>`). Detailed description of this process can be found in [Ciravegna 2000a]. Reliability for contextual rules is computed by using the same error rate used for best rules, but only matches in controlled contexts are counted. In conclusion the sets of tagging rules (LP)² induces are both the best rule pool and the contextual rules. Figure 3 shows the whole algorithm for tagging rule induction.

```

Loop for instance in initial-instances
  unless already-covered(instance)
    loop for rule in generalise(instance)
      test(rule)
      if best-rule?(rule)
        then insert(rule, bestrules)
              cover(rule, initial-instances)
      else loop for tag in tag-list
          if test-in-context(rule,tag,:right)
            then select-contxtl(rule,tag,:right)
          if test-in-context(rule,tag,:left)
            then select-contxtl(rule,tag,:left)

```

Figure 3: The final algorithm for rule tagging induction.

2.2 Inducing Correction Rules

Tagging rules when applied on the test corpus report some imprecision in slot filler boundary detection. A typical mistake is for example “at `<time>` 4 `</time>` pm”, where “pm” should have been part of the time expression. For this reason (LP)² induces rules for shifting wrongly positioned tags to the correct position.

² In the following we just make examples related to the first case as it is more intuitive.

³ The training corpus is used for computing the maximum filler length for each slot.

It learns from the mistakes made in applying tagging rules on the training corpus. Shift rules consider tags misplaced within a distance d from the correct position. Correction rules are identical to tagging rules, but (1) their patterns match also the tags inserted by the tagging rules and (2) their actions shift misplaced tags rather than adding new ones. An example of an initial correction rule for shifting `</stime>` in “at `<stime>` 4 `</stime>` pm” is shown in table 4.

The induction algorithm used for the best tagging rules is also used for shift rules: initial instance identification, generalization, test and selection. “Wrong Tag” and “Correct Tag” conditions are never relaxed. Positive (correct shifts) and negative (wrong shifts of correctly assigned tags) are counted. Shift rules are accepted only if they report an acceptable error rate.

3. Extracting Information

In the testing phase information is extracted from the test corpus in four steps: initial tagging, contextual tagging, correction and validation. The best rule pool is initially used to tag the texts. Then contextual rules are applied in the context of the introduced tags. They are applied until new tags are inserted, i.e. some contextual rules can match also tags inserted by other contextual rules. Then correction rules correct some imprecision. Finally each tag inserted by the algorithm is validated. There is no meaning in producing a start tag (e.g. `<speaker>`) without its corresponding closing tag (`</speaker>`) and vice versa, therefore uncoupled tags are removed in the validation phase.

Condition			Additional Information			
word	Wrong tag	correct tag	lemma	LexCat	case	SemCat
at			at	prep	low	
4	<code></stime></code>		4	digit	low	
pm		<code></stime></code>	pm	other	low	timeid

Table 4: A correction rule. The action (not shown) shifts the tag from the wrong to the correct position.

4. Experimental Results

(LP)² was tested in a number of tasks in two languages: English and Italian. In each experiment (LP)² was trained on a subset of the corpus (some hundreds of texts, depending on the corpus) and the induced rules were tested on unseen texts. Here we report about results on two standard tasks for adaptive IE: the CMU seminar announcements and the Austin job announcements⁴. The first task consists of uniquely identifying speaker name, starting time, ending time and location in 485 seminar announcements [Freitag 1998]. Table 5 shows the overall accuracy obtained by (LP)², and compares it with that obtained by other state of the art algorithms. (LP)² scores the best results in the task. It definitely outperforms other symbolic approaches (+8.7% wrt Rapier[Califf

⁴ Corpora available at www.isi.edu/muslea/RISE/index.html

1998], +21% wrt to Whisk[Soderland 1999]), but it also outperforms statistical approaches (+2.1% wrt BWI [Freitag and Kushmerick 2000] and +4% wrt HMM [Freitag and McCallum 1999]). Moreover (LP)² is the only algorithm whose results never go down 75% on any slot (second best is BWI: 67.7%).

	(LP) ²	BWI	HMM	SRV	Rapier	Whisk
speaker	77.6	67.7	76.6	56.3	53.0	18.3
location	75.0	76.7	78.6	72.3	72.7	66.4
stime	99.0	99.6	98.5	98.5	93.4	92.6
etime	95.5	93.9	62.1	77.9	96.2	86.0
All Slots	86.0	83.9	82.0	77.1	77.3	64.9

Table 5: F-measure ($\beta=1$) obtained on CMU seminars. Results for algorithms other than (LP)² are taken from [Freitag and Kushmerick 2000]. We added the comprehensive ALL SLOTS figure, as it allows better comparison among algorithms. It was computed by:

$$\frac{\sum_{\text{slot}} (\text{F-measure} * \text{number of possible slot fillers})}{\sum_{\text{slot}} \text{number of possible slot fillers}} * 100$$

Concerning (LP)² results from a 10 cross-folder experiment using half of the corpus for training. F-measure calculated via the MUC scorer [Douthat 1998]. Average training time per run: 56 min on a 450MHz computer. Window size $w=4$.

A second task concerned IE from 300 Job Announcements taken from misc.jobs.offered [Califf 1998]. The task consists of identifying for each announcement: message id, job title, salary offered, company offering the job, recruiter, state, city and country where the job is offered, programming language, platform, application area, required and desired years of experience, required and desired degree, and posting date. The results obtained on such a task are reported in table 6. (LP)² outperforms both Rapier and Whisk (Whisk obtained lower accuracy than Rapier [Califf 1998]). We cannot compare (LP)² with BWI as the latter was tested on a very limited subset of slots. In summary, (LP)² reaches the best results on both the tasks.

Slot	(LP) ²	Rapier	BWI	Slot	(LP) ²	Rapier
id	100	97.5	100	platform	80.5	72.5
title	43.9	40.5	50.1	application	78.4	69.3
company	71.9	69.5	78.2	area	66.9	42.4
salary	62.8	67.4		req-years-e	68.8	67.1
recruiter	80.6	68.4		des-years-e	60.4	87.5
state	84.7	90.2		req-degree	84.7	81.5
city	93.0	90.4		des-degree	65.1	72.2
country	81.0	93.2		post date	99.5	99.5
language	91.0	80.6		All Slots	84.1	75.1

Table 6: F-measure ($\beta=1$) obtained on the Jobs domain using half of the corpus for training.

5. Discussion

(LP)²'s main features that are most likely to contribute to the excellence in the experiments are: (1) the induction of symbolic rules (see the conclusions), (2) rule gener-

alization via shallow NLP, (3) the use of single tag rules and (4) the use of correction.

(LP)² induces rules by instance generalization. Generalization is also used in SRV, Rapier and Whisk. It allows reducing data sparseness by capturing some general aspects beyond the simple flat word structure. Shallow NLP is the basis for generalization in (LP)². Morphology allows overcoming data sparseness due to number/gender word realizations, while POS tagging information allows generalization over lexical categories. In principle such type of generalization produces rules of better quality than those matching the flat word sequence, rules that tend to report better effectiveness on unseen cases. This is because both morphology and POS tagging are generic NLP processes performing equally well on unseen cases; therefore rules relying on their results apply successful on unseen cases. This intuition was confirmed experimentally: (LP)² with generalization ((LP)²_G) definitely outperforms a version without generalization ((LP)²_{NG}) on the test corpus, while having comparable results on the training corpus (+57% on the speaker field, +28% on the location field, +11% overall on the CMU task). Moreover in (LP)²_G the covering algorithm converges more rapidly than in (LP)²_{NG}, because its rules tend to cover more cases. This means that (LP)²_G need less examples in order to be trained, i.e., rule generalization also allows reducing the training corpus size. Not surprisingly the role of shallow NLP in the reduction of data sparseness is more relevant on semi-structured or free texts (such as the CMU seminars) than on documents with highly standardized language (e.g. HTML pages, or the job announcement task). During the rule selection phase (LP)² is able to adopt the right level of NLP information for the task at hand: in an experiment on texts written in mixed Italian/English we used an English POS tagger that was completely unreliable on the Italian part of the input. (LP)²_G reached the same effectiveness of (LP)²_{NG}, because the rules using the unreliable NLP information were automatically discarded. This shows that the use of NLP is always a plus, never a minus.

The separate recognition of tags is an aspect shared by BWI, while HHM, Rapier and SRV recognized whole slots and Whisk recognizes multislots. Separate tag identification allows further reduction of data sparseness, as it better generalizes over the coupling of slot start/end conditions. For example in order to learn patterns equivalent to the regular expression (``at'|`starting from``)`DIGIT(`pm'|`am')`, (LP)² just needs two examples, e.g., ``at'+`pm'` and ``starting from'+`am'`, because the algorithm induces two independent rules for `<time>` (``at' + `starting from'`) and two for `</time>` (``am' + `pm'`). In a slot-oriented rule learning strategy four examples (and four rules) will be needed, i.e. ``at'+`pm'`, ``at'+`am'`, ``starting from' +`pm'`, ``starting from'+`am'`. In a multislot approach the problem is worst and the number of training examples needed increases drastically [Ciravegna

2000a].

Another reason for the good experimental results relies in the use of a correction step. Correction is useful in recognizing slots with fillers with high degree of variability (such as the speaker in the CMU experiment), while it does not pay on slots with highly standardized fillers (such as many slots in the Jobs task). (LP)² using correction rules reports 7% more in terms of accuracy on $\langle /speaker \rangle$ than (LP)² without correction. Imprecision in tagging was also reported by [Califf 1998] who noted up to 5% imprecision on some slots (but she did not introduce any correction steps in Rapier).

Slot	PRE	REC	F-measure	Slot	PRE	REC	F-measure
Name	97	82	88.9	Email	92	71	80.1
Street	96	71	81.6	Tel.	93	75	83.0
City	90	90	90	Fax	100	50	66.6
Prov.	97	92	94.4	Zip	100	90	94.7
Zip	100	90	94.7				

Table 7: Results of a blind test on 50 resumes. This is not a simple named entity recognition task. A resume may contain many names and addresses (e.g. previous work addresses, name of referees or thesis supervisors and their addresses). The system had to recognize the correct ones.

6. Developing real world applications

(LP)² was developed as a research prototype, but it quickly turned out to be suitable for real world applications. An industrial system based on (LP)², *LearningPinocchio*, was developed. Recently *LearningPinocchio* has been used in a number of industrial applications. Moreover licenses have been released to external companies for further application development. This section reports about some industrial applications we have directly developed. The system is used for extracting information from professional resumes written in English. It is used on the results of a spider that surfs the Web to retrieve professional resumes. The spider classifies resumes by topics (e.g. computer science). *LearningPinocchio* extracts the relevant information and its output is used to populate a database. Table 7 shows some results obtained in such task. Application development time for the IE task required about 24 person hours for scenario definition and revision (the scenario was refined by tagging some texts in different ways and discussing among annotators). Further 10 person hours were needed for tagging about 250 texts. The rule induction process took 72 hours on a 450MHz machine, with window size $w=4$. Finally system results validation required four person hours.

TAG	F(1)	TAG	F(1)
Geograph Area	0.70	Organiz. Name	0.86
Currency	0.85	Company Share	
Stock Exchange		Name	0.85
Name	0.91	Type	0.92
Index	0.97	Category	0.86
ALL SLOTS 0.87			

Table 8: Results of blind test on financial news (300 texts).

Two other applications were developed for Kataweb, a major Italian Internet portal. The goal was to extract information from both financial news and classified ads written in Italian and published on the portal pages. *LearningPinocchio* is used both to generate hyperlinks for cross-referencing texts and to retrieve texts querying the content. The application is currently under final test at the customer's site. Table 8 shows experimental results on financial news.

7. Conclusions and Future Work

(LP)² is a successful algorithm. On the one hand it outperforms the other state of the art algorithms on two very popular IE tasks. It is important to stress the fact that (LP)² outperforms also statistical approaches, because in the last years the latter largely outperformed symbolic approaches. There is a clear advantage in using symbolic rules in real world applications. It is possible to inspect the final system results and manually add/modify/remove rules for squeezing additional accuracy (it was not done in the scientific experiments, but it was in the applications).

On the other hand (LP)² was the basis for building *LearningPinocchio*, a tool for building adaptive IE applications that is having a considerable commercial success. This shows that adaptive IE is able produce tools suitable for building real world applications by a final user by using only analyst's knowledge.

Future work on (LP)² will involve both the improvement of rule formalism expressiveness and the further use of shallow NLP for generalization. Concerning the improvement in rule formalism expressiveness we plan to include some forms of Kleene-star and optionality operators. Such improvement has shown to be very effective in both BWI and Rapier. Concerning the use of shallow NLP for generalization (i.e., one of the keys of the success in (LP)²) there are two possible improvements. On the one hand (LP)² will be used in cascade with a Named Entity Recognizer (also implemented by using (LP)²). This will allow further generalization over named entity classes (e.g., the speaker is a person, so it is possible to generalize over such class in the rules). On the other hand (LP)² is compatible with forms of shallow parsing such as chunking. It is then possible to preprocess the texts with a chunker and to insert tags only at the chunk borders. This is likely to improve precision in border identification.

An interesting question concerns the limits of the tagging-based IE approach used by many adaptive systems, (LP)² included. Classic MUC-like IE is based on template filling. Template filling is more complex than tagging, as it implies to decide about both coreference of expressions (are "John A. Smith" and "J. Smith" the same person? Two seminars have been identified in a text: are they separate events or are they coreferring?), and slot pairing (two seminars and two speakers have been identified: which is the speaker of the first semi-

nar?). (LP)² is able to apply default strategies for template merging that solve simple cases of coreferences and slot pairing. Such strategies are powerful enough to cope with many real world tasks, but in some other cases they are not effective enough. For example in the resumes application the customer was interested in retrieving also the triples degree/university/year. *LearningPinocchio* was able to correctly highlight such information, but often it was not able to pair them correctly, therefore they were not used to populate the database, but only to index texts. Even if some ad hoc strategies would have probably solved the problem in the specific case, it is quite clear that this is a major limitation in the approach. Classical MUC-like IE systems use sophisticated strategies for coreference resolution and template merging, very often based on a mix of NLP knowledge and domain knowledge [Humphreys *et al.* 1998]. Two problems prevent the use of such techniques. On the one hand deep NLP is not effective in many applications in the Internet realm (e.g. how can you parse an e-mail?). On the other hand it is not clear how to elicit the domain knowledge for coreference from an analyst (adaptability via analyst's knowledge is a strong constraint as mentioned above). Adaptive template filling is an issue worth exploration that we are currently investigating. We work in the direction of further using shallow NLP for improving template filling and merging. To some extent this is also a step in the direction of bridging the gap between classical NLP based IE systems and fully adaptive systems.

Acknowledgments

I developed (LP)² and *LearningPinocchio* at ITC-Irst, Centro per la Ricerca Scientifica e Tecnologica, Trento, Italy. *LearningPinocchio* is property of ITC-Irst, see <http://ecate.itc.it:1025/cirave/LEARNING/home.html>. The financial application mentioned above was jointly developed with Alberto Lavelli. Thanks to Daniela Petrelli for revising this paper. Errors, if any, are mine.

References

- [Califf 1998] Mary E. Califf, Relational Learning Techniques for Natural Language IE, *Ph.D. thesis*, Univ. Texas, Austin, www.cs.utexas.edu/users/mecaliff
- [Cardie 1997] Claire Cardie, 'Empirical methods in information extraction', *AI Journal*, 18(4), 65-79, 1997.
- [Ciravegna *et al.* 2000] Fabio Ciravegna, Alberto Lavelli, and Giorgio Satta, 'Bringing information extraction out of the labs: the Pinocchio Environment', in *ECAI2000, Proc. of the 14th European Conference on Artificial Intelligence*, ed., W. Horn, Amsterdam, 2000. IOS Press.
- [Ciravegna 2000a] Fabio Ciravegna, 'Learning to Tag for Information Extraction from Text' in F. Ciravegna, R. Basili, R. Gaizauskas (eds.) *ECAI Workshop on Machine Learning for Information Extraction*, Berlin, August 2000. (www.dcs.shef.ac.uk/~fabio/ecai-workshop.html)
- [Douthat 1998] Aaron Douthat, 'The message understanding conference scoring software user's manual', in *the 7th Message Understanding Conf.*, www.muc.saic.com
- [Freitag 1998] Dayne Freitag, 'Information Extraction from HTML: Application of a general learning approach', *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 1998.
- [Freitag and McCallum 1999] Dayne Freitag and Andrew McCallum: 'Information Extraction with HMMs and Shrinkage', *AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, FL, 1999, www.isi.edu/~muslea/RISE/ML4IE/
- [Freitag and Kushmerick 2000] Dayne Freitag and Nicholas Kushmerick, 'Boosted wrapper induction', in F. Ciravegna, R. Basili, R. Gaizauskas (eds.) *ECAI2000 Workshop on Machine Learning for Information Extraction*, Berlin, 2000, (www.dcs.shef.ac.uk/~fabio/ecai-workshop.html)
- [Grishman 1997] Ralph Grishman, 'Information Extraction: Techniques and Challenges', In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, in M.T. Pazienza, (ed.), Springer, 97.
- [Humphreys *et al.* 1998] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, Y. Wilks: 'Description of the University of Sheffield LaSIE-II System as used for MUC-7'. In *Proc. of the 7th Message Understanding Conference*, 1998 (www.muc.saic.com).
- [Kushmerick *et al.* 1997] N. Kushmerick, D. Weld, and R. Doorenbos, 'Wrapper induction for information extraction', *Proc. of 15th International Conference on Artificial Intelligence, IJCAI-97*, 1997.
- [Miller *et al.* 1998] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone and R. Weischedel, 'BBN: Description of the SIFT system as used for MUC-7', In *Proc. of the 7th Message Understanding Conference*, 1998 (www.muc.saic.com).
- [Muslea *et al.* 1998] I. Muslea, S. Minton, and C. Knoblock, 'Wrapper induction for semi-structured, web-based information sources', in *Proc. of the Conference on Autonomous Learning and Discovery CONALD-98*, 1998.
- [Soderland 1999] Steven Soderland, 'Learning information extraction rules for semi-structured and free text', *Machine Learning*, (1), 1-44, 1999.
- [Yangarber *et al.* 2000] Roman Yangarber, Ralph Grishman, Pasi Tapanainen and Silja Huttunen: "Automatic Acquisition of Domain Knowledge for Information Extraction" In *Proc. of COLING 2000, 18th Intern. Conference on Computational Linguistics*, Saarbrücken, 2000.

Relational Learning via Propositional Algorithms: An Information Extraction Case Study*

Dan Roth

Wen-tau Yih

Department of Computer Science
University of Illinois at Urbana-Champaign
{danr, yih}@uiuc.edu

Abstract

This paper develops a new paradigm for relational learning which allows for the representation and learning of relational information using propositional means. This paradigm suggests different tradeoffs than those in the traditional approach to this problem – the ILP approach – and as a result it enjoys several significant advantages over it. In particular, the new paradigm is more flexible and allows the use of any propositional algorithm, including probabilistic algorithms, within it.

We evaluate the new approach on an important and relation-intensive task - Information Extraction - and show that it outperforms existing methods while being orders of magnitude more efficient.

1 Introduction

Relational learning is the problem of learning structured concept definitions from structured examples. Relational representations use a first-order model to describe the problem domain and examples are given to the learner in terms of objects and object relations rather than by simple propositions.

In a variety of AI problems such as natural language understanding related tasks, visual interpretation and planning, given a collection of objects along with some relations that hold among them, the fundamental problem is to learn definitions for some relations or concepts of interest in terms of the given relations. Examples include the problem of identifying noun phrases in a sentence in terms of the information in the sentence, detecting faces in an image or defining a policy that maps states and goals to actions in a planning situation. In many of these cases it is natural to represent and learn concepts relationally; propositional representations might be too large, could lose much of the inherent domain structure and consequently might not generalize well. In recent years, this realization has renewed the interest in studying relational representations and learning.

Inductive Logic Programming (ILP) is an active subfield of machine learning that addresses relational learning and is a natural approach to apply to these tasks. While, in principle,

ILP methods could allow induction over relational structures and unbounded data structures, theoretical and practical considerations render the use of unrestricted ILP methods impossible. Studies in ILP suggest that unless the rule representation is severely restricted the learning problem is intractable. While there are several successful heuristics for learning ILP programs, there are many practical difficulties with the inflexibility, brittleness and inefficient “generic” ILP systems. In most cases, researchers had to develop their own, problem specific, ILP systems [Mooney, 1997] but have not always escaped problems such as search control and inefficiency - especially in large scale domains like NLP.

This paper develops a different paradigm for relational learning that allows the use of general purpose and efficient propositional algorithms, but nevertheless learns relational representations. Our paradigm takes a fresh look at some of the restrictions ILP systems must make in order to work in practice and suggests alternatives to these; as a result, it enjoys several significant advantages over traditional approaches. In particular, the new paradigm is more flexible and allows the use of any propositional algorithm, including probabilistic algorithms, within it. It maintains the advantages of ILP approaches while allowing more efficient learning, improved expressivity and robustness.

At the center of our paradigm is a knowledge representation language that allows one to efficiently represent and evaluate rich relational structures using propositional representations. This allows us to learn using propositional algorithms but results in relational concepts descriptions as outcome.

This paper evaluates the new paradigm in the domain of Information Extraction (IE). This is the NLP task of extracting specific types or relevant items from unrestricted text. Relational learning methods are especially appealing for learning in this domain since both the target concepts and the information in the domain (within and across documents) are often relational. We develop an IE system based on our paradigm; the learning component of our system makes use of a feature efficient learning algorithm that is especially suitable for the nature of our paradigm. However, we show that standard learning algorithms like naive Bayes also work well with it. Experimental comparisons show that our approach outperforms several ILP-based systems tried on this tasks, sometime significantly, while being orders of magnitude more efficient.

*Research supported by NSF grants IIS-9801638 and IIS-0085836 and an ONR MURI Award.

2 Propositional Relational Representations

In this section we present a knowledge representation language that has two components: (1) a subset of first order logic (FOL) and (2) a collection of structures (graphs) defined over elements in the domain.

The relational language \mathcal{R} is a restricted (function free) first order language for representing knowledge with respect to a domain \mathcal{D} . The restrictions on \mathcal{R} are applied by limiting the formulae allowed in the language to a collection of formulae that can be evaluated very efficiently on given instances (interpretations). This is done by (1) defining primitive formulae with limited scope of the quantifiers (Def. 2.1). (2) General formulae are defined inductively in terms of primitive formulae in a restricted way that depends on the relational structures in the domain. The emphasis is on locality with respect to these relational structures that are represented as graphs over the domain elements.

This language allows the encoding of first order representations and relational structures as propositions and thus supports the use of better learning algorithms, including general purpose propositional algorithms and probabilistic algorithms over the elements of the language. This approach extends previous related constructions from ILP [Lavrac *et al.*, 1991; Khardon *et al.*, 1999] but technically is more related to the latter. In the rest of this section we present the main constructs of the language. We omit many of the standard definitions and concentrates on the unique characteristics of \mathcal{R} . See, e.g., [Lloyd, 1987] for general details.

The vocabulary Σ consists of constants, variables, predicate symbols, quantifiers, and connectives.

Definition 2.1 A primitive formula is defined inductively: (1) A term is either a variable or a constant. (2) Let p be a k -ary predicate, t_1, \dots, t_k terms. Then $p(t_1, \dots, t_k)$ is an atomic formula. (3) Let F be an atomic formula, and z be a variable. Then $(\forall z F)$ and $(\exists z F)$ are atomic formulae. (4) An atomic formula is a primitive formula. (5) If F and G are primitive formulae, then so are $(\neg F)$, $(F \wedge G)$, $(F \vee G)$.

Notice that for primitive formulae in \mathcal{R} the scope of a quantifier is always the unique predicate that occurs with it in the atomic formula. We call a variable-less atomic formula a *proposition* and a quantified atomic formula, a *quantified proposition* [Khardon *et al.*, 1999]. The informal semantics of the quantifiers and connectives is as usual.

Before we move on to extend the language \mathcal{R} we discuss the domain and the notion of an instance.

Definition 2.2 (Domain) The language \mathcal{R} is defined over a domain \mathcal{D} which consists of a structured element $D = \langle \mathcal{V}, \mathcal{G} \rangle$ where \mathcal{V} is a collection of typed elements and \mathcal{G} is a set of partial orders over \mathcal{V} (specifically, each partial order $g_i \in \mathcal{G}$ is an acyclic graph (V_i, E_i) , where $V_i \subseteq \mathcal{V}$ and E_i is a set of edges on V_i). Along with it, for each constant there is an assignment of an element in \mathcal{V} and for each k -ary predicate, an assignment of a mapping from V^k to $\{0, 1\}$ ($\{true, false\}$).

Example 2.1 The fragment in Fig. 1 forms a domain $\mathcal{D} = \langle \mathcal{V}, \mathcal{G} \rangle$, where \mathcal{V} consists words **TIME**, **:**, **3**, **:**, **30**, **pm** and the phrase **3 : 30 pm** and \mathcal{G} consists of two linked lists (denoted in the solid line and the dashed line).

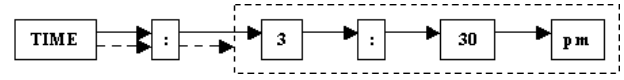


Figure 1: A fragment of an article

could be more complex and represent, say, a parse tree over a sentence.

The notion of types is used as a way to classify elements in the language according to their properties. In particular, we will think also of predicates as typed in the sense that their domain is typed. We distinguish within the set \mathcal{V} two main types; a set $\mathcal{O} \subset \mathcal{V}$ of *objects* and a set $\mathcal{A} \subset \mathcal{V}$ of *attributes*. Correspondingly, we define types of predicates. Type 1 predicates take as their first argument an element in \mathcal{O} and as their second argument an element in \mathcal{A} . Type 1 predicates describe properties of elements in \mathcal{O} ; thus $p(o, a)$ may also be written as $p(o) = a$, with the semantics that in a given interpretation, $p(o, a)$ holds. All other predicate types will have elements in \mathcal{O} as their arguments. These predicates will be defined via elements in \mathcal{G} , and will have $g \in \mathcal{G}$ as their type. Specifically, $p_g(o_1, o_2)$ indicates that o_1, o_2 are nodes in the graph $g \in \mathcal{G}$ and there is an edge in g between them.

Elements in \mathcal{O} represent objects in the world. For example, in NLP applications such as ours these might be words, phrases, sentences or documents. Predicates of type 1 describe properties of these elements – spelling of a word, syntactic tag of a word, a phrase type, etc. The graphs in \mathcal{G} describe relations between (sets of) objects – word w_1 is *before* w_2 , w_1 is the *subject* of the verb w_2 , etc. (It is possible to generalize \mathcal{G} to a collection of hypergraphs, but this is not needed in the current application.) As usual, the “world” in which we interpret the aforementioned constructs could be a single sentence, a document, etc.

Definition 2.3 An instance is an interpretation [Lloyd, 1987] which lists a set of domain elements and the truth values of all instantiations of the predicates on them.

Given an instance x , a formula F in \mathcal{R} is given a unique truth value, the value of F on x , defined inductively using the truth values of the predicates in F and the semantics of the connectives. Since for primitive formulae in \mathcal{R} the scope of a quantifier is always the unique predicate that occurs with it in the atomic formula, we have the following properties. It will be clear that the way we extend the language (Sec. 2.1) maintains this properties (proofs omitted).

Proposition 2.1 Let F be a formula in \mathcal{R} , let x be an instance, and let t_p be the time to evaluate the truth value of an atom p in F . Then, the value of F on x can be evaluated in time $\sum_{p \in F} t_p$.

That is, F is evaluated simply by evaluating each of its atoms (ground or quantified) separately. This holds, similarly, for the following version of subsumption for formulae in \mathcal{R} .

Proposition 2.2 (subsumption) Let x be an instance and let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function of n variables that can be evaluated in time t_f . Then the value of the clause $f(F_1, \dots, F_n)$ on x can be evaluated in time $t_f + \sum_F t_F$, where the sum is over all n formulae that are arguments of f .

2.1 Relation Generation Functions

The definition of the general formulae allowed in \mathcal{R} will be operational and will use the structures in \mathcal{G} . To do that we introduce a mechanism that generates more expressive formulae in a way that respects the structures in the domain, thus restricting the formulae generated.

Definition 2.4 A formula in \mathcal{R} maps an instance x to its truth value in x . It is active in x if it has truth value true in it. We denote by X the set of all instances – the instance space. A formula $F \in \mathcal{R}$ is thus a relation over X , $F : X \rightarrow \{0, 1\}$.

Example 2.2 Let instance x be the fragment illustrated in Ex. 2.1. Some active relations in x are `word(TIME)`, `word(pm)`, and `number(30)`.

Given an instance, we would like to know what are the relations (formulae) that are active in it. We would like to do that, though, without the need to write down explicitly all possible formulae in the domain. This is important, in particular, over infinite domains or in problems domains such as NLP, where inactive relations vastly outnumber active relations.

Definition 2.5 Let \mathcal{X} be an enumerable collection of relations on X . A relation generation function (RGF) is a mapping $G : X \rightarrow 2^{\mathcal{X}}$ that maps $x \in X$ to a set of all elements in \mathcal{X} that satisfy $\chi(x) = 1$. If there is no $\chi \in \mathcal{X}$ for which $\chi(x) = 1$, $G(x) = \phi$.

RGFs can be thought of as a way to define “kinds” of formulae, or to parameterize over a large space of formulae. Only when an instance x is presented, a concrete formula (or a collection of) is generated. An RGF can be thought of as having its own range \mathcal{X} of relations.

Example 2.3 It is impossible to list all formulae that use the number predicate in advance. However, RGF can specify formulae of this kind and, given the instance `TIME : 3 : 30 pm`, only the active relations of this kind: `number(3)` and `number(30)` – are generated.

In order to define the collection of formulae in \mathcal{R} we define the family of RGFs for \mathcal{R} ; the output of these define the formulae in \mathcal{R} . RGFs are defined inductively using a relational calculus. The alphabet of this calculus consists of (i) basic RGFs, called *sensors* and (ii) a set of connectives. While the connectives are the same for every alphabet the *sensors* vary from domain to domain. A sensor is a way to encode basic information one can extract from an instance. It can also be used as a uniform way to incorporate external knowledge sources that aid in extracting information from an instance.

Definition 2.6 A sensor is a relation generation function that maps an instance x into a set of atomic formulae in \mathcal{R} . When evaluated on an instance x a sensor s outputs all atomic formulae in its range which are active.

Example 2.4 Following are some sensors that are commonly used in NLP.

- The word sensor over word elements, which outputs active relations `word(TIME)`, `word(·)`, `word(3)`, `word(30)`, and `word(pm)` from “`TIME : 3 : 30 pm`”.
- The length sensor over phrase elements, which outputs active relations `len(4)` from “`3 : 30 pm`”.

- The is-a sensor, outputs the semantic class of a word.
- The tag sensor, outputs the part-of-speech tag of a word

The word and len sensors derive information directly from the raw data, while the is-a sensor uses external information sources such as WordNet and the tag sensor uses a pre-learned part-of-speech tagger.

Several mechanisms are used in the relational calculus to define the operations of RGFs. We mention here only the *focus* mechanism which is actually a *binding* mechanism that is used to define quantified formulae in \mathcal{R} .

Definition 2.7 Let E be a set of elements in the domain. An RGF r is focused on E if, given an instance x , it generates only formulae in its range that are active in x due to elements in E . The focused RGF is denoted $r[E]$.

There are several ways to define a focus set. It can be done explicitly or using the structure \mathcal{G} . The focus set is equivalent to the free variables in FOL representations.

The relational calculus allows one to inductively generate new RGFs by applying connective and quantifiers over existing RGFs. Using the standard connectives one can define RGFs that output formulae of the type defined in Def 2.1. Details will not be given here. Instead, we describe only one important type of operations within the relational calculus – structural operations. These operations exploit the structural (relational) properties of the domain as expressed in \mathcal{G} in order to define RGFs. Thus, more general formulae, that can have interactions between variables, are generated, while still allowing for efficient evaluation and subsumption, due to the graph structure. For example, the structural collocation operator, *colloc*, with respect to g is defined as follows.

Definition 2.8 Let s_1, s_2, \dots, s_k be RGFs for \mathcal{R} . $colloc_g(s_1, s_2, \dots, s_k)$ is a restricted conjunctive operator that is evaluated on a chain of length k in g . Specifically, let $\mathcal{D} = (\mathcal{V}, \mathcal{G})$ be a domain, with $g \in \mathcal{G}$, and v_1, v_2, \dots, v_k a chain in g . The formulae generated by $colloc_g(s_1, s_2, \dots, s_k)$ are those generated by $s_1[v_1] \& s_2[v_2] \& \dots \& s_k[v_k]$, where (1) by $s_j[v_j]$ we mean that the RGF s_j is focused on $\{v_j\}$ (2) the $\&$ operator means that formulae in the output of $(s \& r)$ are active formulae of the form $F \wedge G$, where F is in the range of s and G is in the range of r (evaluated on x). This is needed since each RGF in the conjunction may produce more than one formulae.

Example 2.5 When applied with respect to the graph g which represents the linear structure of the sentence, $colloc_g$ simply generates formulae that corresponds to ngrams. E.g., given the fragment “Dr John Smith”, RGF $colloc(\text{word}, \text{word})$ extracts the bigrams `word(Dr) - word(John)` and `word(John) - word(Smith)`.

Similarly to $colloc_g$ one can define a *sparse* collocation operator with respect to a chain in g . This is also a restricted conjunctive operator that is evaluated on a chain in g with the following difference. Formulae are generated by $scolloc_g(s_1, s_2, \dots, s_k)$ as follows: Let v_1, v_2, \dots, v_n be a chain in g . For each subset $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ of elements in v , such that $i_j < i_l$ when $j < l$, all the formulae: $s_1[v_{i_1}] \& s_2[v_{i_2}] \& \dots \& s_k[v_{i_k}]$, are generated.

Notice that while primitive formulae in \mathcal{R} have a single predicate in their scope, the structural properties provides a way to go beyond that but only in a restricted way that is efficiently evaluated. Structural operations allow us to define RGFs that constrain formulae evaluated on different objects without incurring the cost usually associated with enlarging the scope of free variables. This is done by enlarging the scope only as required by the structure of the domain, modeled by \mathcal{G} . This allows for efficient evaluation as in Prop. 2.1, 2.2 with the only additional cost being that of finding chain in the graph (details omitted).

3 Comparison to ILP Methods

Propositional learning on relational features provides a different paradigm for relational learning. While, in principle, ILP methods could allow induction over relational structures and unbounded data structures and their expressivity cannot be matched by propositional methods, theoretical and practical considerations render the use of unrestricted ILP methods impossible. Studies in ILP suggest that, unless the rule representation is severely restricted, the learning problem is intractable [Kietz and Dzeroski, 1994; Cohen, 1995; Cohen and Page, 1995]. Several successful heuristics for learning ILP programs [Muggleton and De Raedt, 1994; Cussens, 1997; Quinlan, 1990] have made different algorithmic and representational restrictions in order to facilitate ILP although there are still many practical difficulties with the inflexibility, brittleness and inefficient “generic” ILP systems.

Our approach offers different tradeoffs than those suggested by the common restrictions made by current ILP systems. While we cannot say that our paradigm dominates the traditional ILP approach in general, we believe that in many cases the alternatives it offers provide a better way to address relational learning, especially in large scale domains such as NLP. Below we address some key issues that could help in developing a better understanding to the suitability of each.

Search: The key difference between the traditional ILP approach and ours is the way they structure the search space. In ILP, “features” are generated as part of the search procedure in an attempt to find good bindings. In our case, the “features” tried by an ILP program during its search are generated up front (in a data driven way) by the RGFs. Some of these are grounded and some have free variables in them. The learning algorithm will look at all of them “in parallel” and find the best representation. Search control methods used by ILP methods are thus analogous to the expressivity we give to our RGFs. The order of “visiting” these features is different.

Knowledge: One of the key cited advantages of ILP methods is the ability to incorporate background knowledge. In our paradigm, this is incorporated flexibly using the notion of *sensors*. Sensors allow us to treat information that is readily available in the input, external information or even previously learned concepts in a uniform way.

Expressivity(I): The basic building blocks of the representations we use (our formulae) are the same as those used by ILP representations. As presented in Sec. 2, for a predicate R and elements a, b we are not representing only the ground term $R(a, b)$ but also $R(X, Y)$, $R(X, b)$, etc. This is similar to the

work in [Lavrac *et al.*, 1991] only that our structural operations allow us to avoid some of the determinacy problems of that approach.

Learning: The relational features generated by our RGFs provide a uniform domain for different learning algorithms. Applying different algorithms is easy and straightforward. Moreover, it is straightforward to use probabilistic models over this representation and in this way it provides a natural and general way of using relational representations within a probabilistic framework. On the other hand, it turns out that “generic” ILP methods suffer brittleness and inefficiency and in many cases, researchers had to develop their own, problem specific, ILP systems [Mooney, 1997].

In particular, while time complexity is a significant problem for ILP methods, propositional learning is typically a lot more efficient. In our paradigm, due to the fact that our RGFs will generate a very large number of relational features (see “search” above) we adopt a specific learning methodology, following [Khardon *et al.*, 1999]. While this is not necessary (as shown in Sec. 5) we discuss this direction next.

Expressivity (II): Several issues can be mentioned in the context of using linear threshold functions as concept representation over the relational features extracted using RGFs [Khardon *et al.*, 1999]. Advocates of ILP methods suggest that the rich expressive power of FOL provides advantages for knowledge-intensive problems such as NLP [Mooney, 1997]. However, given strong intractability results, practical systems apply many representational restrictions. In particular, the depth of the clauses (the number of predicates in each clause) is severely restricted. Thus, the learned concept is actually a k -DNF, for small k . In our paradigm, the constructs of *colloc* and *scolloc* allow us to generate relational features which are conjunctions of predicates and are thus similar to a clause in the output representation of an ILP program. While an ILP program represents a disjunction over these, a linear threshold function over these relational features is more expressive. In this way, it may allow learning smaller programs. The following example illustrates the representational issues:

Example 3.1 Assume that in several seminar announcements, fragments that represent speaker have the pattern:
 $\cdots \text{Speaker} : \text{Dr FName LName line-feed} \cdots$

An ILP rule for extracting speaker could then be:
 $\text{before_tag}(2, \text{“Speaker”}) \wedge \text{contains}(\text{target}, \text{“Dr”}) \wedge \text{after_tag}(1, \text{line-feed}) \rightarrow \text{speaker}(\text{target})$

That is, the second word before the target phrase is “Speaker”, target phrase contains word “Dr”, and the “line-feed” character is right after the target phrase. In our relational feature space all the elements of this rule (and many others) would be features, but the above conjunction is also a feature. Therefore a collection of clauses of this form becomes a disjunction in our feature space and will be learned efficiently using a linear threshold element.

Finally, we mention that for learning linear threshold elements there exist feature efficient algorithms [Littlestone, 1988] that are suitable for learning in NLP-like domains, where the number of potential features is very large, but only a few of them are active in each example, and only a small

fraction of them are relevant to the target concept.

4 Case Study – Information Extraction

Information Extraction (IE) is a natural language processing (NLP) task that processes unrestricted text and attempts to extract specific types of items from the text.

This form of shallow text processing has attracted considerable attention recently with the growing need to intelligently process the huge amounts of information available in the form of text documents. While learning methods have been used earlier to aid in parts of an IE system [Riloff, 1993; Soderland and Lehnert, 1994], it has been argued quite convincingly [Califf and Mooney, 1999; Craven and Slattery, 2001] that relational methods are necessary in order to learn how to directly extract the desired items from documents. The reason is that the target concepts require the representation of relations over the source document and learning those might require induction over structured examples and FOL representations. Indeed, previous works [Califf and Mooney, 1999; Freitag, 2000] have demonstrated the success of ILP methods in this domain. This is therefore an ideal domain to study our proposed relational learning paradigm.

4.1 Problem Description

In this paper, the IE task is defined as locating specific fragments of an article according to predefined slots in a template. Each article is a plain-text document that consists of a sequence of tokens. Specifically, the data used in experiments is a set of 485 seminar announcements from CMU¹. The goal is to extract four types of fragments from each article² – those describing the start time (*stime*) and end time (*etime*) of the seminar, its location (*location*) and the seminar’s speaker (*speaker*). Given an article, our system picks at most one fragment for one slot. If this fragment represents the slot, then it is a correct prediction. Otherwise, it is a wrong prediction (including the case that the article doesn’t contain the slot at all) [Freitag, 2000].

4.2 Extracting Relational Features

The basic strategy of our IE solution is to learn a classifier that discriminates a specific desired fragment. First, we generate examples for this type of fragment. This is done by:

1. Identifying candidate fragments in the document. (All fragments are candidates; in training, fragments are annotated.) Note: fragments may overlap, and only a small number of them contain desired information.
2. For each candidate fragment, use the defined RGF features to re-represent it as an example which consists of all active features extracted for this fragment.

Let $f = (t_i, t_{i+1}, \dots, t_j)$ be a fragment, with t_i representing tokens and $i, i+1, \dots, j$ are positions of tokens

¹The data set was originally collected from newsgroups and annotated by Dayne Freitag; it is available on <http://www.isi.edu/~muslea/RISE/>.

²An article might not contain one of the fields, e.g., *etime*, or might contain one of them, e.g., the speaker, more than once.

in the document. Our RGFs are defined to extract features from three regions: left window $(t_{i-w}, \dots, t_{i-1})$, target fragment (t_i, \dots, t_j) , and right window $(t_{j+1}, \dots, t_{j+w})$, where w is the window size.

The domain formed by these three regions contains two types of elements – word elements (e.g., t_{i-w}, \dots, t_{j+w}) and one phrase element (the target region). The RGFs are focused either on a specific word element (one free variable) or on the borders of the phrase element (two free variables) and define relational features relative to these. In the next section we provide examples of RGFs used in experiments.

4.3 Two-stage Architecture

Once examples are generated, the IE task is accomplished by two learning stages. The same classifier, SNoW, is used in both stages, but in a slightly different way.

SNoW [Roth, 1998; Carleson *et al.*, 1999] is a multi-class classifier that is specifically tailored for large scale learning tasks. The SNoW learning architecture learns a sparse network of linear functions, in which the targets (fragment types, in this case) are represented as linear functions over a common feature space. SNoW has already been used successfully for a variety of tasks in natural language and visual processing [Golding and Roth, 1999; Roth *et al.*, 2000]. SNoW is built on a feature efficient learning algorithm, Winnow [Littlestone, 1988] and therefore is an ideal learning approach to complement our paradigm, as discussed before. While SNoW can be used as a classifier and predicts using a winner-take-all mechanism over the activation values of the target classes, here we rely directly on the activation value it outputs, computed using a sigmoid function over the linear sum. The normalized activation value can be shown to be a distribution function and we rely heavily on its robustness in our two-stage architecture. The two stages are (1) Filtering: reduce the amount of candidates from all possible fragments to a small number, and (2) Classifying: pick the right fragment from the preserved fragments by the learned classifier. Theoretical justification for this architecture will be presented in a companion paper. Intuitively, this architecture increases the expressivity of our classification system. Moreover, eliminating most of the negative examples significantly reduces the number of irrelevant features, an important issue given the small data set.

Filtering: One common property of IE tasks is that negative examples (irrelevant fragments) extremely outnumber positive examples (fragments that represent legitimate slots). In the seminar announcements data set, for example, 0.3% of the fragments represent legitimate slots. This stage attempts to filter out most of the negative examples without eliminating positive examples. It can also be viewed as a classifier designed to achieve high recall, while the classifier in the second stage aims at high precision. The filter consists of two learned classifiers; a fragment is filtered out if it meets one of the following criteria:

1. Single feature classifier: Fragment doesn’t contain a feature that should be active in positive examples.
2. General Classifier: Fragment’s confidence value is below the threshold.

System	stime			etime			loc			speaker		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
SNoW-IE	99.6	99.6	99.6	97.6	95.0	96.3	90.9	64.1	75.2	83.3	66.3	73.8
NB-IE	98.3	98.3	98.3	96.5	92.8	94.6	76.8	62.0	68.6	40.3	32.0	35.7
RAPIER-WT	96.5	95.3	95.9	94.9	94.4	94.6	91.0	61.5	73.4	79.0	40.0	53.1
RAPIER	93.9	92.9	93.4	95.8	94.6	95.2	91.0	60.5	72.7	80.9	39.4	53.0
SRV	98.6	98.4	98.5	67.3	92.6	77.9	74.5	70.1	72.2	54.4	58.4	56.3
WHISK	86.2	100.0	92.6	85.0	87.2	86.1	83.6	55.4	66.6	52.6	11.1	18.3

Table 1: Results for seminar announcements task

For criterion 1, it turns out that there exists some features that are (almost) always active in positive examples. For example, in our experiments, the length of fragments satisfies this: $[len(fragment) \leq 7]$ is always satisfied by stime fragments. Also, $[fragment \text{ contains a word that is a noun}]$ always holds in speaker fragments.

For criterion 2, implemented using SNoW, relying on its robust confidence estimation, the problem becomes finding the right threshold. Minimum activation values of positive examples in training data are used as thresholds (for the different types of slots). Examples with lower activation values are filtered out.

The two stages also differ in the RGFs used. The following, more crude RGFs are used at the filtering stage:

- Target region: *word*, *tag*, *word&tag*, *colloc(word, word)*, *colloc(word, tag)*, *colloc(tag, word)*, *colloc(tag, tag)* on word elements, and *len* on the phrase element.
- Left & Right window: *word&loc*, *tag&loc*, and *word&tag&loc*, where *loc* extracts the position of the word in the window.

Classifying: Fragments that survived the filtering stage are then classified using a second SNoW classifier, for the four slots. First, an additional collection of RGFs is applied to enhance the representation of the candidate fragments, thus allowing for more accurate classification. As before, in training, the remaining fragments are annotated and are used as positive or negative examples to train the classifiers. In testing, the remaining fragments are evaluated on the learned classifiers to determine if they can fill one of the desired slots. In this stage 4 different classifiers are trained, one for each type of fragments. All examples are run through all 4 classifiers. The RGFs added in this stage include:

For *etime* and *stime*:

`scolloc[word&loc(-1)[l.window], word&loc[r.window]],`
`scolloc[word&loc(-1)[l.window], tag&loc[r.window]]`

For *location* and *speaker*:

`scolloc[word&loc(-2)[l.window], tag[tag], tag&loc(1)[r.window]]`

The first set of RGFs is a sparse structural conjunction of the word directly left of the target region, and of words and tags in the right window (with relative positions). The second is a sparse structural conjunction of the last two words in the left window, a tag in the target, and the first tag in the right window.

A decision is made by each of the 4 classifiers. A fragment is classified as type *t* if the *t*th classifier decides so. At most one fragment of type *t* is chosen in each article, based on the activation value of the corresponding classifier.

slot	pass(training)	pass(testing)	loss(testing)
stime	4.75%	4.72%	0.94%
etime	1.07%	1.07%	1.64%
location	15.42%	15.30%	3.28%
speaker	14.89%	14.28%	2.79%

Table 2: Filtering efficiency

5 Experimental Results

Our experiments use the same data, methodology and evaluation metrics used by several ILP-based IE systems in previous works. The systems such as RAPIER [Califf and Mooney, 1999], SRV [Freitag, 2000], and WHISK [Soderland, 1999] have also been tested on this data set. The data (485 documents) is randomly split into two sets of equal sizes, one for training and the other for testing. The reported results are an average of five runs. As usual, the performance is quantified in terms of *Precision* (*P*) – the percentage of correct predictions – and *Recall* (*R*) – the percentage of *slots* that are identified. We also report the $F1 = (\frac{PR}{P+R})$. The results of our system, SNoW-IE, are shown in the first row of table 1 along with the results of several other ILP-based IE systems that were tested on this task under the same conditions. An exception is WHISK, for which the results are from a 10-fold validation using only 100 documents randomly selected from the training set. The systems also use somewhat different information sources. The words in the documents are used by all systems. Part-of-speech tags are used both in RAPIER-WT and SNoW-IE; SRV uses other predicates that capture POS information to some extent. A version of RAPIER uses also semantic information; this can be done in our system by adding, say, an is-a sensor but, given their results we did not incorporate this information.

In addition to our SNoW-IE we have also experimented with a second propositional algorithm, the naive Bayes (NB-IE) algorithm. NB was used on exactly the same set of features (same examples) that were generated using our relational paradigm for SNoW, and in exactly the same way. Although the results of NB-IE are not as good as those of SNoW-IE – the quality of the classifier is certainly an important issue – the experiments with a second propositional algorithm exhibit the fact that our relational paradigm is a general one. As indicated in [Craven and Slattery, 2001; Freitag, 2000] a simple minded use of this algorithm is not competitive for this task; but on top of a paradigm that is able to exploit the relational nature of the data it compares favor-

ably with ILP methods. Overall, SNoW-IE outperforms the existing rule-based IE systems on all the four slots.

To clarify, we note that the output representation of our system makes use of similar type of relational features as do the ILP-based systems, only that instead of a collection of conjunctive rules over these, it is represented as a linear function.

It is difficult to isolate the contribution of our two-stage architecture to the quality of the results. We believe, though, that the ease of incorporating this and other learning architectures is an indication to the flexibility of the approach and the advantages of learning with propositional means. Table 2 gives some insight into that, by showing the average performance of the filtering stage. The first two columns show the ratio of training and testing examples that pass the filter. The third column lists the ratio of positive examples in the testing set that are filtered out. These fragments do not even reach the second stage. (Since an article may contain more than one fragment that represents the same slot, it is sometimes still possible for the classifier to pick the correct slot.)

6 Conclusion

The use of relational methods is back in fashion. It became clear that for a variety of AI problems there is a fundamental need to learn and represent relations and concepts in terms of other relations. Information Extraction – the task of extracting relevant items from unrestricted text is one such task.

This paper suggests a new paradigm for relational learning – which allows for the representation and learning of relational information using propositional means. We argue that our paradigm has different tradeoffs than the traditional approach to this problem – the ILP approach – and as a result it enjoys several significant advantages over it. In particular, the suggested paradigm is more flexible and allows the use of any propositional algorithm within it, including probabilistic approaches. As such, it addresses in a natural and general way the problem of using relational representations within a probabilistic framework, an important problem which has been studied a lot recently.

Our paradigm is exemplified on an important task - Information Extraction (IE). Based on our paradigm, we developed a new approach to learning for IE, and have shown that it outperforms existing ILP-based methods. Moreover, it is several orders of magnitude more efficient. We believe that this work opens up several directions for further work – on relational learning and knowledge representation and on practical and efficient solutions to NLP and IE problems.

References

- [Califf and Mooney, 1999] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *National Conference on Artificial Intelligence*, 1999.
- [Carleson *et al.*, 1999] A. Carleson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC CS Dept., May 1999.
- [Cohen and Page, 1995] W. Cohen and D. Page. Polynomial learnability and inductive logic programming: Methods and results. *New Generation Computing*, pages 369–409, 1995.
- [Cohen, 1995] W. Cohen. PAC-learning recursive logic programs: Negative result. *Journal of Artificial Intelligence Research*, 2:541–573, 1995.
- [Craven and Slattery, 2001] M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43:97–119, 2001.
- [Cussens, 1997] J. Cussens. Part-of-speech tagging using progol. In *International Workshop on Inductive Logic Programming*, pages 93–108, Prague, Czech Republic, 1997. Springer. LNAI 1297.
- [Freitag, 2000] D. Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, 2000.
- [Golding and Roth, 1999] A. R. Golding and D. Roth. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999.
- [Khardon *et al.*, 1999] R. Khardon, D. Roth, and L. G. Valiant. Relational learning for NLP using linear threshold elements. In *Proc. of the International Joint Conference of Artificial Intelligence*, pages 911–917, 1999.
- [Kietz and Dzeroski, 1994] J. Kietz and S. Dzeroski. Inductive logic programming and learnability. *SIGART Bulletin*, 5(1):22–32, 1994.
- [Lavrac *et al.*, 1991] N. Lavrac, S. Dzeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *Machine Learning (EWSL-91)*, volume 482 of *LNAI*, pages 265–281, Porto, Portugal, 1991. Springer Verlag.
- [Littlestone, 1988] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Programming*. Springer-verlag, 1987.
- [Mooney, 1997] Raymond J. Mooney. Inductive logic programming for natural language processing. In *(ILP-96)*, volume 1314 of *LNAI*, pages 3–24. Springer, 1997.
- [Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 20:629–679, 1994.
- [Quinlan, 1990] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
- [Riloff, 1993] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *National Conference on Artificial Intelligence*, pages 811–816, 1993.
- [Roth *et al.*, 2000] D. Roth, M-H. Yang, and N. Ahuja. Learning to recognize objects. In *CVPR'00*, 2000.
- [Roth, 1998] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *National Conference on Artificial Intelligence*, pages 806–813, 1998.
- [Soderland and Lehnert, 1994] S. Soderland and W. Lehnert. Wrap-up: a trainable discourse module for information extraction. *Journal of Artificial Intelligence Research*, 2:131–158, 1994.
- [Soderland, 1999] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

Deriving a multi-domain information extraction system from a rough ontology

Thierry Poibeau

Thales Research and Technology Laboratoire d'Informatique de Paris-Nord
Domaine de Corbeville, F-91404 Orsay av. J.-B. Clément, F-93430 Villetaneuse France
Thierry.Poibeau@thalesgroup.com

Abstract

This paper presents a multi-domain information extraction system. In order to decrease the time spent on the elaboration of resources for the IE system and guide the end-user in a new domain, we suggest to use a machine learning system that helps defining new templates and associated resources. This knowledge is automatically derived from the text collection, in interaction with the end-user to rapidly develop a local ontology giving an accurate image of the content of the text. The system is finally evaluated using classical indicators.

1 Introduction

Information Extraction (IE) is a technology dedicated to the extraction of structured information from texts. This technique is used to highlight relevant sequences in the original text or to fill pre-defined templates [Pazienza, 1997]. Below is the example of a story concerning a terrorist attack in Spain:

22 août 1990, page 24

ESPAGNE un mort au cours d'un attentat à la voiture piégée au Pays Basque.

- Une personne a été tuée mardi 21 août en milieu de journée à Oyarzun (province basque de Guipuzcoa) au cours de l'explosion d'une voiture piégée dans le parking d'un hypermarché, a indiqué la police. (AFP.)¹

¹ This is the translation of the text:

August 22th, 1990, page 24

SPAIN: one person killed in a booby-trapped car attack in the Basque Country

One person was killed on Tuesday August 21st during the day in Oyarzun (Basque province from Guipuzcoa) in the explosion of a booby-trapped car on the parking of a supermarket, indicated the police. (AFP)

and the corresponding entry in the database filled by the IE system.

Event date: 22 août 1990

Event location: Espagne, Pays Basque

Number of killed people: 1

Number of injured people: 0

Weapon: voiture piégée

Even if IE seems to be now a relatively mature technology, it suffers from a number of yet unsolved problems that limit its dissemination through industrial applications. Among these limitations, we can consider the fact that systems are not really portable from one domain to another. Even if the system is using some generic components, most of its knowledge resources are domain-dependent. Moving from one domain to another means re-developing some resources, which is a boring and time-consuming task (for example Riloff [1995] mentions a 1500 hours development).

Moreover, when information is often changing (think of the analysis of a newswire for example), one might want to elaborate new extraction templates. This task is rarely addressed by the research studies in IE system adaptation, but we noticed that it is not an obvious problem. People are not aware of what they can expect from an IE system, and most of the time they have no idea of how deriving a template from a collection of texts can be. On the other hand, if they defined a template, the task cannot be performed because they are waiting for information that is not contained in the texts.

In order to decrease the time spent on the elaboration of resources for the IE system and guide the end-user in a new domain, we suggest to use a machine learning system that helps defining new templates and associated resources. This knowledge is automatically derived from the text collection, in interaction with the end-user to rapidly develop a local ontology giving an accurate image of the content of the text. The experiment also aims at reaching a better coverage thanks to the generalization process provided by the machine learning system.

We will firstly present the overall system architecture and principles. The learning system is then what allows the learning of semantic knowledge to help define templates for new domains. We will show to what extent it is possible to speed up the elaboration of resources without any decrease in the quality of the system. We will finish with some comments on this experiment and we will show how domain-specific knowledge acquired by the learning system such as the subcategorization frame of verbs could be used to extract more precise information from texts.

2 System architecture and principles

The system can be divided into three main parts:

1. A machine learning engine used to produce semantic clusters from the corpus. These clusters are weighted and are intended to give to the expert a rough idea of the topic addressed in the text;
2. A system to help the creation of extraction template once the relevant topic of the corpus have been identified;
3. The information extraction system itself, that will use the resources defined at the previous stage to fill the templates.

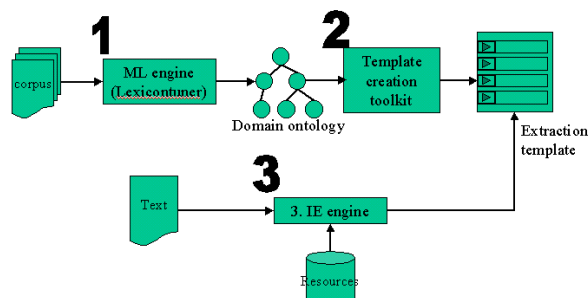


Figure 1: Architecture of the system

The above schema gives an overview of the overall architecture. The corpus is processed by the machine learning system (1), in order to produce semantic clusters organized in a superficial ontology. The template creation module (2) helps the expert define his own extraction template from the ontology. The lower part of the schema describes the information extraction system itself (3), processing a text to fill the extraction template.

The information extraction system consists in a multi-agent platform. Each agent performs a precise subtask of the information extraction process (named entity recognition, relation analysis, template filling). A supervisor controls the overall process and the information flow (for more detail about the system architecture, see [Poibeau 2001]).

Apart from the information extraction system itself, the machine learning module is intended to help the end-user produce the extraction template. A representative set of documents has to be processed by the learning module to obtain an ontology and some rough resources for the

domain he wants to cover. The resources have to be manually completed to obtain a good coverage of the domain.

3 Extraction of Semantic Clusters from Lexical Resources

The acquisition method combines knowledge contained in on-line resources and statistical data obtained from the training corpus of the chosen domain. This method is implemented through the LexiconTuner, which was initially developed for French but can be used for any language if correct resources are provided. The method is inspired from [Luk 1995] and described in details in [Ecran 1996] and [Poibeau 1999].

Semantic clusters – that is to say clusters of semantically related words – can be acquired from on-line resources including dictionaries, thesauri and taxonomies [Wilks *et al.*, 1995]. Dictionaries model the meanings of lexical items using textual definitions. Textual definitions are written in natural language and the full information encoded in these definitions can't be extracted easily. However, partial information, which can be used as semantic clusters, can be extracted from the definitions relatively easily.

The idea of building networks from definitions was first proposed by Véronis and Ide [1990], along with a propagation technique for computing clusters of related words. Many authors have proposed techniques for deriving some kind of clusters or associations of concepts from graphs of concepts, among others transitive closures and computation of graph “cliques”, simulated annealing, etc. Some of these experiments derived information from a dictionary; see for example the early work at New Mexico State University [Fowler and Dearholt, 1989]. In our approach, the content words in the definitions of a word sense are used as component concepts of the word sense [Luk, 1995], that is to say that a concept is a lexical item considered as a member of a semantic cluster. For example, the word *opera*, which is defined as “dramatic performance or composition of which music is an essential part...” in the *Concise Oxford Dictionary*, can be assigned the semantic concepts of dramatic, performance, composition and music.

In the LexiconTuner, the generation of the clusters is carried out in two steps. Firstly, the list of all the concepts occurring in the corpus is generated. For each word in the corpus, the program looks for its textual definition(s) in the lexicon. The words in the definitions are treated as concepts and are recorded. This step allows us to go from a French word (e.g. *opéra*) to some concepts labeled with English words (dramatic, performance, composition, music), by means of a bilingual dictionary (*opera* being defined as a “dramatic performance or composition of which music is an essential part”).

Secondly, semantic clusters are generated by clustering related concepts in the list. The program uses a semantic net

as its semantic knowledge source to determine the semantic relatedness between the concepts. The net is derived from the *Oxford Advanced Learner's Dictionary* (OALD). Each node represents a word, and for any two words $w1$ and $w2$, $w1$ is linked to $w2$ if $w2$ appears in the definitions of $w1$ in OALD (*music* will be linked to *composition* if *music* appears in the definition of *composition*). The link from $w1$ to $w2$ is weighted inversely proportionally to the product of the number of senses of $w1$ as defined in OALD and the number of words in the definition of $w1$ that contains $w2$. The clusters are formed in 3 steps:

- Every circle in the semantic net which length is equal to a user-defined number is identified. For example, if the number is 3, the system will generate all the sets consisting of three related words according to the dictionary definition. For *opera*, the following associations will be considered: (*opera*, *performance*, *dramatic*) and (*dramatic*, *composition*, *music*).
- Circles that share one or more nodes are merged. Thus, (*opera*, *performance*, *dramatic*) and (*dramatic*, *composition*, *music*) are merged into a single set (*opera*, *performance*, *dramatic*, *composition*, *music*). This is called a "core cluster".
- Lastly, peripheral words (words which are part of a circle which length is inferior to the user-defined number) are related to the core cluster. If two words like *dramatic* and *comic* are related, then *comic* will be added to the cluster, being related to *dramatic*.

This algorithm does not really capture all the complexity of dictionary definitions. For example, although cycles are common among dictionary definition terms, there may also be chains of words which are highly related but which do not form a cycle. To solve this problem, [Lesk, 1986] suggested a method to cluster word pairs based on "degree of overlap" between their head words or definition words, in the context of word sense disambiguation. In our system, the merging step and the addition of peripheral words partially allow taking into account chains of words.

Lastly, the membership of a cluster is "weighed". The weight of a core member is considered as the inverse of the number of senses of the word as defined in OALD. The weight of a non-core member is considered as the mean of the weights of the core members of the cluster to which it is related. Then, semantic clusters can capture notions appearing in texts independently from the lexical items expressing these notions.

4 Automatically deriving a rough ontology from the corpus

Not all semantic clusters are equally useful for Information Extraction in any particular domain. The usefulness of a semantic cluster can be determined statistically from the corpus itself. [Salton, 1988] suggested measures that reveal

the usefulness of a term: applied to the members of a cluster, these measures reveal the usefulness of a semantic cluster in the corpus [Resnik, 1995] [Aguirre & Rigau, 1996].

The IE prototype for French implemented a method to measure the usefulness of semantic clusters using a single function which captures the variance of the occurrence frequency [Luk, 1995]. The variance of the occurrence frequency of a semantic cluster s is given as:

$$Var(f(s,A)) = E[(f(s,A) - E(f(s,A)))^2]$$

where $f(s,A)$ is the occurrence frequency of s in a collection of articles A , and $E(X)$ is the mean of X for any X .

The measurement captures both the discriminatory power and the relevancy of a semantic cluster: if the occurrence frequency of a semantic cluster has a high variance, then it must be both discriminative and relevant. If a semantic cluster has low relevancy (i.e. the cluster has low occurrence frequency in most of the articles), then the variance of occurrence frequency will be low relatively to a cluster with a higher frequency. Meanwhile, if a semantic cluster has low discriminatory power (i.e. the semantic cluster has similar occurrence frequency in most articles), then the variance of occurrence frequency will also be low. Lastly, it also takes into account the dependency of the discriminatory power on the occurrence frequencies, normalized according to the size of the document. The weight of a cluster is then normalized according to the size of the document using the following function:

$$Wn(f,d) = W(f,d) * \frac{1}{\log(N(d)/200) + 1}$$

if $N(d) \geq 200$ and

$$Wn(f,d) = W(f,d) * \log(200/N(d)) + 1$$

Otherwise.

where $Wn(f,d)$ and $W(f,d)$ are the normalized and non-normalized weights of the feature f in document d , and $N(d)$ is the number of content words in document d . The use of a logarithmic expression allows to smooth the result in spite of the differences in the size of the documents. The constant 200 is the average size of a document in the corpus, without taking into account empty words.

After a rough discrimination between the more useful semantic clusters and the less useful ones obtained by these statistical methods, domain experts can manually refine the selections of the semantic clusters, which are to be used in the Information Extraction system. Lastly a set of representative clusters is associated to each text. These clusters have a higher weight in the text than in the corpus.

5 Template design

Semantic classes produced by the LexiconTuner are proposed to the end-user, who chooses which clusters are of

interest to him. Once he has chosen one cluster, the system automatically proposes him an interface to refine the cluster, aggregate a part of the closest clusters and develop a hierarchy. This hierarchy can be assimilated to a local ontology, describing a part of the world knowledge related to the event of interest for the end-user.

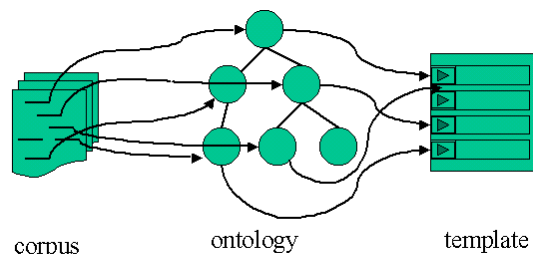


Figure 2: From corpus to template, through the ontology...

The semantic clusters produced by the LexiconTuner give to the expert a rough idea of the topics addressed in the corpus. The expert has to navigate among these clusters to select relevant topics and establish a candidate template. The chosen topics has to be named and to be associated with a slot in the template. The system automatically generates the corresponding information into the database: the creation of a new template leads to the creation of a new table and each new slot in the template corresponds to a new column in the table. This technical part of the template creation process can be compared to the Tabula Rasa toolkit developed at New Mexico State University to help end-users define their own templates [Ogden and Bernick, 1997].

The evaluation of the template creation task is not obvious since it necessitates domain knowledge and text manipulation from the experts. No clear reference can be established. The only way to evaluate the contribution of the semantic clusters is to ask the expert firstly to manually elaborate templates from a corpus and secondly to do it with the help of the LexiconTuner. The expert we worked with made the following comments:

- The semantic clusters produced by the LexiconTuner give an appropriate idea of the overall topic addressed by the texts.
- These clusters help the elaboration of templates and allows to focus on some part of information without reading large part of texts.
- However, the elaboration of the template itself remains largely dependant of the domain knowledge of the expert because (a) he knows what kind of information he wants to find in relation with a given topic and (b) the clusters are too coarse-grain to directly correspond to slots.

When the template corresponds to an event, the information to be found generally refers to classical *wh-questions*: who, what, where and when. Some additional slots can be added but most of the time they correspond to classical

associations of ideas. For example, if one wants to extract information about football matches, he will immediately create a slot corresponding to the score. However, this comment is due to the fact that the system is analyzing news stories, for which one can associate stereotypic reduced templates (sometimes called *templettes* in the IE community).

Date	28-ja-00	Text
Location	PARIS	
Personality	Lionel Jospin	
Type	Déclaration	
Source	AFP (wire)	
Topic	- Lionel Jospin a assuré vendredi que le gouvernement entendait "rénover l'épargne salariale", selon un communiqué de ses services.	

Figure 3: A specific reduced template (a "templette")

We observed that the template creation task is frequently a problem in more technical domains for which no clear *a priori* schema exists. In this experiment, the LexiconTuner can be seen as a tool to explore the corpus and give a rough idea of tentative templates rather than a tool designed to help the creation of the content of the templates themselves. However, the LexiconTuner results is also useful for the creation of the resources of the system.

6 Experiment description and evaluation

We asked an expert to define a new template and the associated resources, using the tools we presented above. We chose the terrorist event domain from the AFP newswire, because it is a well-established task since the MUC-4 and MUC-5 conferences. Moreover, similar experiments has been previously done that give a good point of reference (see [Poibea 1999] and [Faure and Poibea 2000]).

6.1 Resources design process

Once a new template is designed, the developer has to elaborate resources for this template. For a more detailed description of this process (resource design for a single template), please refer to [Faure and Poibea 2000]. We only describe here the main steps of this process.

Homogeneous semantic clusters learned by the LexiconTuner are refined: a manual work of the expert is necessary to exploit semantic classes (merging of scattered classes, deletion of irrelevant elements, addition of new elements, etc.). About five hours have been dedicated, after the acquisition process, to the refinement of data furnished by LexiconTuner. Merging and structuring classes incrementally develop a local ontology, which nodes are related to slots in the extraction template. This knowledge is

also considered as a resource for the finite-state system (Intex cf. [Silberstein, 1993]) and is exploited either as dictionaries or as transducers, according to the nature of the information. If it is a general information that is not domain specific, the development guidelines advise the user to use dictionaries that can be reused, otherwise, he designs a transducer.

A dictionary is a list of words or phrases, each one being accompanied by a tag and a list of features. The first names dictionary or the location dictionary are generic reusable resources. Below is a sample of the location names dictionary²:

```
Abidjan, .N+Loc+City
Afghanistan, .N+Loc+Country
Allemagne, .N+Loc+Country
Allemagne de l'Ouest, .N+Loc+Country
Allemagne de l'Est, .N+Loc+Country
```

These items structured in a list are convenient for the dictionary format and the semantic lists elaborated from the LexiconTuner complete in an accurate manner the coverage of the initial dictionaries from Intex.

The transducer format is essentially used for more complex or more variable data where linguistic phenomena such as insertion or optionality may interfere. The figure 4 is the illustration of a transducer recognizing *blessé par l'explosion de Det N (explosion of Det N, injured by the explosion of...)*, where the nominal phrase *Det N* recognizes nominal phases elaborated from the semantic class bombing where the following words appear: *bombe, obus, grenade*, etc (*bomb, shell, grenade...*).

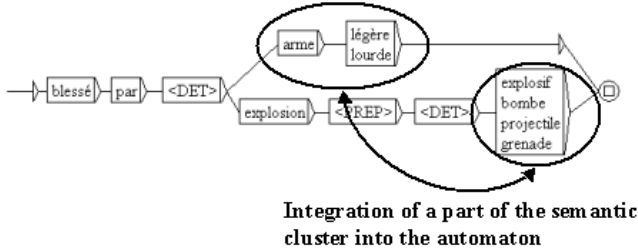


Figure 4: “weapon” automaton

The elaboration of such transducers requires linguistic expertise to obtain *in fine* a system recognizing the relevant sequences without too much noise. Some tools and a method have been defined for the semi-acquisition and the design of resources such as subcategorization frames, completing the clusters learned by the LexiconTuner [Poibeau, 2000]. The architecture of the system is using cascading transducers, it is then important that each level has a good quality in order to allow the following analysis level to operate on a solid background. The different

² Each line begins with a term, followed by some indication about its syntactic category *N* for *Noun* and semantic features *Loc* to indicate a location, *Country* to indicate a country, etc.

transducers are then minimized and determined³. The overall set of transducers is composed of 1000 nodes and about 5000 arrows in our experiment.

6.2 Evaluation

A hundred texts have been used as “training corpus” and a hundred different texts have been used as “test corpus”. Texts are first parsed with our system, and then some heuristics allow to fill the extraction template: the first occurrence of a number of victims or injured persons is stored. If a text deals with more than one terrorist event, we assume that only the first one is relevant. Thanks to the nature of the channel, very few texts deal with more than one event.

Our results have been evaluated by two human experts who did not follow our experiment. Let *Pos* be the total number of good answers and *Act* the number of solutions proposed by the system. Our performance indicators are defined as:

- (Ok) if extracted information is correct;
- (False) if extracted information is incorrect or not filled;
- (None) if there were no extracted information and no information has to be extracted.

Using these indicators, we can compute two different values:

- Precision (*Prec*) is defined as Ok/Pos .
- Recall (*Rec*) is defined as $Ok/(Act)$.

Slot name	Precision	Recall	P&R
Event date	.95	.96	.95
Event location	.88	.92	.90
Nb of killed people	.83	.73	.77
Nb of injured people	.80	.69	.74
Weapon	.85	.82	.83
Total	.86	.82	.83

The performances are good according to the state-of-the-art and to the time spent on resource development. However, we can analyze the remaining errors as follows:

The date of the story is nearly fully correct because the wrapper uses the formatted structure of the article to extract it. The errors for the location slot are due to two “contradictory” locations found by the system. A more complete linguistic analysis or a database providing lists of cities in different countries would reduce this kind of errors. The errors in the number of dead or injured persons slot are frequently due to silence: for example the system fails against too complex syntactic forms. The silence for the weapon slot is frequently due to incompleteness of semantic dictionaries.

³ These two operations allow to optimize the analysis time.

7 Conclusion and future work

The experiment that has been described is based on an external knowledge base derived from a dictionary. It is thus different from [Faure and Poibeau 2000] which tries to acquire knowledge directly from the text. The use of an external database allows to work on middle-size corpora that are not as redundant as technical texts. We also think that using a general dictionary is interesting when dealing with general texts like a newswire. Clusters contain words that were not contained in the training part of the corpus, allowing a better coverage of the final result.

The multi-domain extraction system is currently running in real time, on the AFP newswire. About 15 templates have been defined that cover about 30% of the stories. From the remaining 70%, the system only extract surface information, especially thanks to the wrappers. The performances are between .55 and .85 P&R, if we do not take into account the date and location slots that are filled by means of wrappers. New extraction templates are defined to prove system scalability (about one new template per week). We hope to reach the number 50 templates towards summer 2001.

8 Acknowledgement

A part of this study re-used some pieces of software developed in the framework of the ECRAN project (LE-2110, 1996-1999). I would like to thank Alpha Luk and other people having participated to the development of the Lexicon Tuner. I am also indebted to David Faure, Tristelle Kervel, Adeline Nazarenko and Claire Nedellec for useful comments and discussions on this subject.

9 References

[Aguirre and Rigau, 1996] Aguirre E. and Rigau G. Word Sense Disambiguation using conceptual density. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Copenhagen, 1996.

[Ecran, 1996] D-2.5.1 - Methods for Lexical Items Modification/Creation and D-2.6.1 - Heuristics for Automatic Tuning. ECRAN Project Deliverable, 1996.

[Faure and Poibeau, 2000] Faure D. and Poibeau T. First experiments of using semantic knowledge learned by Asium for Information Extraction task using Intex. In *Proceedings of the workshop on learning ontologies*, during *ECAI'2000*, Berlin, 2000.

[Fowler and Dearholt, 1989] Fowler R.H. and Dearholt, D.W. Pathfinder networks in information retrieval. Las Cruces, New Mexico State University (Memorandums in Computer and Cognitive Science, M CCS-89-147). 1989.

[Lesk, 1986] Lesk M. Automatic Sense Disambiguation: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of 1986 ACM SIGDOC Conference*, New York. 1986. ACM.

[Luk, 1995] Luk A. K. Statistical Sense Disambiguation with Relatively Small Corpora using Dictionary Definitions. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 1995.

[Pazienza, 1997] Pazienza M. T. (éd.). *Information extraction (a multidisciplinary approach to an emerging information technology)*, Springer Verlag (Lecture Notes in Computer Science), Heidelberg, Germany, 1997.

[Ogden and Bernick, 1997] Ogden W. and Bernick P. Tabula Rasa Meta-Tool: Text Extraction Toolbuilder Toolkit. Technical Report M CCS-97-305. Las Cruces: Computing Research Laboratory. 1997.

[Poibeau 1999] Poibeau T. A statistical clustering method to provide a semantic indexing of texts. In *Workshop on machine learning for information filtering*, during *IJCAI'1999*, Stockholm, 1999.

[Poibeau 2000] Poibeau T. Corpus-based learning for Information Extraction. In *Proceeding of the workshop Machine Learning for Information Extraction*, during *ECAI'2000*, Berlin. 2000.

[Poibeau 2001] Poibeau T. An open architecture for multi-domain information extraction. In *Proceeding of the 13th Innovative Application of Artificial Intelligence*, Seattle, 2001. AAAI Press.

[Resnik, 1995] Resnik Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, 1995.

[Riloff, 1995] Riloff E. Little Words Can Make a Big Difference for Text Classification. In *Proceedings of the 18th Annual International Conference on research and Development in Information Retrieval (SIGIR'95)*, 1995.

[Salton, 1988] Salton G. *Automatic Text Processing*. Addison-Wesley, Reading, MA.

[Silberztein 1993] Silberztein M. (1993) Dictionnaires électroniques et analyse automatique des textes. Masson, Paris.

[Véronis and Ide, 1990] Véronis J. and Ide N. M. Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, Helsinki, Finland.

[Wilks et al., 1995] Wilks Y., Slator B. and Guthrie L. *Electric words: dictionaries, computers and meanings*, MIT Press, Cambridge, MA.

NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

INFORMATION EXTRACTION AND RETRIEVAL

Representing Sentence Structure in Hidden Markov Models for Information Extraction

Soumya Ray^{*†}
sray@cs.wisc.edu

Mark Craven^{†*}
craven@biostat.wisc.edu

^{*}Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706

[†]Department of Biostatistics & Medical Informatics
University of Wisconsin
Madison, Wisconsin 53706

Abstract

We study the application of Hidden Markov Models (HMMs) to learning information extractors for n -ary relations from free text. We propose an approach to representing the grammatical structure of sentences in the states of the model. We also investigate using an objective function during HMM training which maximizes the ability of the learned models to identify the phrases of interest. We evaluate our methods by deriving extractors for two binary relations in biomedical domains. Our experiments indicate that our approach learns more accurate models than several baseline approaches.

1 Introduction

Information extraction (IE) may be defined as the task of automatically extracting instances of specified classes or relations from text. In our research, we are interested in using machine learning approaches, including hidden Markov models (HMMs), to extract certain relationships among objects from biomedical text sources. We present and evaluate two contributions to the state of the art in learning information extractors with HMMs. First, we investigate an approach to incorporating information about the grammatical structure of sentences into HMM architectures. Second, we investigate an objective function for HMM training whose emphasis is on maximizing the ability of the learned models to identify the phrases of interest rather than simply maximizing the likelihood of the training data. Our experiments in two challenging real world domains indicate that both contributions lead to more accurate learned models.

Automated methods for information extraction have several valuable applications including populating knowledge bases and databases, summarizing collections of documents, and identifying significant but unknown relationships among objects. Since constructing information extraction systems manually has proven to be expensive[Riloff, 1996], there has been much recent interest in using machine learning methods to learn information extraction models from labeled training data. Hidden Markov models are among the more successful approaches considered for learning information extractors [Leek, 1997; Freitag and McCallum, 1999;

Seymore *et al.*, 1999; Freitag and McCallum, 2000; McCallum *et al.*, 2000].

Previous HMM approaches to information extraction do not adequately address several key aspects of the problem domains on which we are focused. First, the data we are processing is complex natural language text. Whereas previous approaches have represented their data as sequences of tokens, we present an approach in which sentences are first processed by a shallow parser and then represented as sequences of typed phrases. Second, the data we are processing include many sentences that are not relevant to the relations of interest. Even in relevant sentences, only certain phrases contain information to be extracted. Whereas previous approaches to applying HMMs for IE have focused the training process on maximizing the likelihood of the training sentences, we adopt a training method that is designed to maximize the probability of assigning the correct labels to various parts of the sentences being processed [Krogh, 1994]. Our approach involves coupling the algorithm devised by Krogh with the use of *null models* which are intended to represent data not directly relevant to the task at hand.

2 Problem Domain

Our work is focused on extracting instances of specific relations of interest from abstracts in the MEDLINE database [National Library of Medicine, 2001]. MEDLINE contains bibliographic information and abstracts from more than 4,000 biomedical journals.

An example of a binary relation that we consider in our experiments is the *subcellular-localization* relation, which represents the location of a particular protein within a cell. We refer to the *domains* of this relation as PROTEIN and LOCATION. We refer to an instance of a relation as a *tuple*. Figure 1 provides an illustration of our extraction task. The top of the figure shows two sentences in a MEDLINE abstract. The bottom of the figure shows the instance of the target relation *subcellular-localization* that we would like to extract from the second sentence. This tuple asserts that the protein UBC6 is found in the subcellular compartment called the endoplasmic reticulum.

In order to learn models to perform this task, training examples consisting of passages of text, annotated with the tuples that should be extracted from them, are needed. In our

“...
Here we report the identification of an integral membrane ubiquitin-conjugating enzyme. This enzyme, UBC6, localizes to the endoplasmic reticulum, with the catalytic domain facing the cytosol.
...”

↓
subcellular-localization(UBC6,endoplasmic
reticulum)

Figure 1: An example of the information extraction task. The top shows part of a document from which we wish to extract instances of the subcellular-localization relation. The bottom shows the extracted tuple.

approach, each training and test instance is an individual sentence. There are several aspects of the data that make this a difficult information extraction task: (i) it involves free text, (ii) the genre of text is, in general, not grammatically simple, (iii) the text includes a lot of technical terminology, (iv) there are many sentences from which nothing should be extracted.

In the terminology that has been used in the information extraction literature, our task is inherently a *multiple slot* extraction task. Since we are interested in extracting instances of n -ary relations, we cannot treat each domain of such a relation as a separate unary component to be extracted (also called *single slot* extraction). Consider the subcellular-localization relation discussed above. A document may mention many proteins and many locations but this relation holds only among certain pairs of these proteins and locations.

In the experiments reported here, we use two data sets representing two different binary relations. The subcellular-localization data set includes 545 sentences that represent positive instances (labeled with tuples that should be extracted from them) and 6,700 sentences that represent negative instances (not labeled with any tuples). The 545 positive instances are labeled with 645 tuples in all; there are 335 unique tuples. The second data set is for a binary relation that characterizes associations between genes and genetic disorders. We refer to this relation as disorder-association, and the domains of this relation as GENE and DISORDER. This data set contains 892 positive instances and 11,487 negative instances. The positive instances are labeled with 899 tuples in all (126 unique). For both data sets, the negative instances are “near misses” in that they come from the same population of abstracts as the positive instances, and in many cases they discuss concepts that are associated with the target relation.

The target tuples for the subcellular-localization relation were collected from the Yeast Protein Database (YPD) [Hodges *et al.*, 1998], and the target tuples for the disorder-association relation were collected from the Online Mendelian Inheritance in Man (OMIM) database [Center for Medical Genetics, 2001]. Relevant MEDLINE abstracts were also gathered from entries in these databases. To label the sentences in these abstracts, we matched the target tuples to the words in the sentence. A sentence which contained words that matched a tuple was taken to be a positive in-

stance. Every other sentence was considered to be a negative instance. It is clear that while this process is automatic, it will result in a noisy labeling. A sentence may have the words in a target tuple of the relation while the semantics may not refer to the relation. On the other hand, a tuple in a relation may be described by synonymous words which were not in any target tuple; therefore, sentences where tuples exist as synonyms are labeled incorrectly. We used a random sample of 200 positive and 200 negative sentences to estimate the amount of noise introduced by the labeling process. We estimate with 95% confidence that approximately 10% to 15% of the sentences are labeled incorrectly (either falsely labeled or unlabeled when they should have been) in the subcellular-localization data set. We believe that the disorder-association data set is not as noisy as the subcellular-localization data set.

3 Representing Phrase Structure

Hidden Markov Models (HMMs) are the stochastic analogs of finite state automata. An HMM is defined by a set of states and a set of transitions between them. Each state has an associated emission distribution which defines the likelihood of a state to emit various tokens. The transitions from a given state have an associated transition distribution which defines the likelihood of the next state given the current state.

In previous HMM approaches to information extraction, sentences have been represented as sequences of tokens. We hypothesize that incorporating sentence structure into the models we build results in better extraction accuracy.

Our approach is based on using syntactic parses of all sentences we process. In particular, we use the Sundance system [Riloff, 1998] to obtain a shallow parse of each given sentence. Our representation does not incorporate all of the information provided by a Sundance parse, but instead “flattens” it into a sequence of phrase segments. Each phrase segment consists of a *type* describing the grammatical nature of the phrase, and the words that are part of the phrase.

In positive training examples, if a segment contains a word or words that belong to a domain in a target tuple, it is annotated with the corresponding domain. We refer to these annotations as *labels*. Labels are absent in the test instances. Figure 2a shows a sentence containing an instance of the subcellular-localization relation and its annotated segments (we shall discuss the other panels of this figure later). The second phrase segment in this example is a noun phrase segment (NP_SEGMENT) that contains the protein name UBC6 (hence the PROTEIN label). Note that the types are constants that are pre-defined by our representation of Sundance parses, while the labels are defined with respect to the domains of relation we are trying to extract. Also note that the parsing is not always accurate, for instance, the third segment in figure 2a should really be a VP_SEGMENT, but has been typed as an NP_SEGMENT by Sundance.

The states in our HMMs represent the annotated segments of a sentence. Like a segment, each state in the model is annotated with a $\langle type, label \rangle$ pair¹. A given state can emit only segments whose type is identical to the state’s type; for

¹We can think of segments that do not have a label corresponding to a domain of the relation as having an implicit *empty* label.

“This enzyme, UBC6, localizes to the endoplasmic reticulum, with the catalytic domain facing the cytosol.”					
NP_SEGMENT	this enzyme	DET	this		this
NP_SEGMENT:PROTEIN	ubc6	UNK	enzyme		enzyme
NP_SEGMENT	localizes	UNK:PROTEIN	ubc6	PROTEIN	ubc6
PP_SEGMENT	to	UNK	localizes		localizes
NP_SEGMENT:LOCATION	the endoplasmic reticulum	PREP	to		to
PP_SEGMENT	with	ART	the		the
NP_SEGMENT	the catalytic domain	N:LOCATION	endoplasmic	LOCATION	endoplasmic
VP_SEGMENT	facing	UNK:LOCATION	reticulum	LOCATION	reticulum
NP_SEGMENT	the cytosol	PREP	with		with
		ART	the		the
		N	catalytic		catalytic
		UNK	domain		domain
		V	facing		facing
		ART	the		the
		N	cytosol		cytosol
(a)		(b)			(c)

Figure 2: HMM input representations. (a) The *Phrase* representation: the sentence is segmented into typed phrases. (b) The *POS* representation: the sentence is segmented into words typed with part-of-speech tags. (c) The *Token* representation: the sentence is segmented into untyped words. For each representation, the labels (PROTEIN, LOCATION) are only present in the training sentences.

example, the segment “this enzyme” in figure 2a could be emitted by any state with type NP_SEGMENT, regardless of its label. Each state that has a label corresponding to a domain of the relation plays a direct role in extracting tuples.

Figure 3 is a schematic of the architecture of our phrase-based hidden Markov models. The top of the figure shows the *positive model*, which is trained to represent positive instances in the training set. The bottom of the figure shows the *null model*, which is trained to represent negative instances in the training set. Since our *Phrase* representation includes 14 phrase types, both models have 14 states without labels, and the positive model also has five to six additional labeled states (one for each $\langle \text{type}, \text{label} \rangle$ combination that occurs in the training set). We assume a fully connected model, that is, the model may emit a segment of any type at any position within a sentence.

To train and test our *Phrase* models, we have to modify the standard Forward, Backward and Viterbi algorithms [Rabiner, 1989]. The Forward algorithm calculates the probability $\alpha_q(i)$ of a sentence being in state q of the model after having emitted i elements of an instance. When a sentence is represented as a sequence of tokens, the algorithm is based on the following recurrence:

$$\begin{aligned}
 \alpha_{START}(0) &= 1 \\
 \alpha_q(0) &= 0, q \neq START \\
 \alpha_q(i) &= M(w_i|q) \sum_r T(q|r) \alpha_r(i-1) \quad (1)
 \end{aligned}$$

where M and T represent the emission and transition distributions respectively, w_i is the i^{th} element in the instance, and r ranges over the states that transition to q .

Our modification involves changing the last part of this re-

currence as follows:

$$\alpha_q(i) = \begin{cases} \left(\prod_{j=1}^{|p_i|} M(w_j|q) \right) \sum_r T(q|r) \alpha_r(i-1), & \text{if } \text{type}(q) = \text{type}(p_i); \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Here w_j is the j^{th} word in the i^{th} phrase segment p_i , and type is a function that returns the *type* of a segment or state as described above. The two key aspects of this modification are that (i) the type of a segment has to agree with the type of state in order for the state to emit it, and (ii) the emission probability of the words in the segment is computed as the product of the emission probabilities of the individual words. This latter aspect is analogous to having states use a Naïve Bayes model for the words in a phrase. Note that this equation requires a normalization factor to define a proper distribution over sentences. However, since we use these equations to make relative comparisons only, we leave this factor implicit. The modifications to the Viterbi and Backward algorithms are similar to this modification of the Forward algorithm.

Given these modifications to the Forward and Backward algorithm, we could train phrase-based models using the Baum-Welch algorithm [Baum, 1972]. However, for the models we consider here, there is no hidden state for training examples (i.e., there is an unambiguous path through the model for each example), and thus there is no need to use Baum-Welch. Instead, we assume a fully connected model and obtain transition frequencies by considering how often segments with various $\langle \text{type}, \text{label} \rangle$ annotations are adjacent to each other. We smooth these frequencies over the set of possible transitions for every state using m -estimates [Cestnik, 1990]. In a similar manner, we obtain the emission frequencies of the words in each state by summing over all segments with the

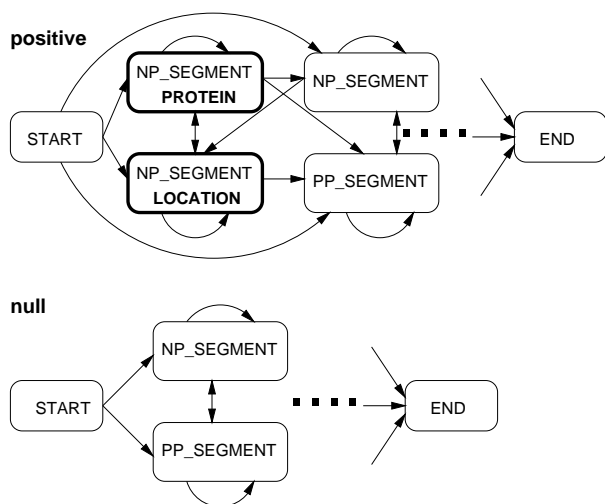


Figure 3: The general architecture of our phrase-based HMMs. The top part of the figure shows the *positive model* and the bottom part of the figure shows the *null model*.

same $\langle \text{type}, \text{label} \rangle$ annotations in our training set. We smooth these frequency counts using m -estimates over the entire vocabulary of words.

Once the model has been constructed, we use it to predict tuples in test sentences. We use the Viterbi algorithm, modified as described above, to determine the most likely path of a sentence through the positive model. We consider a sentence to represent a tuple of the target relation if and only if two conditions hold:

1. The likelihood of emission of the sentence by the positive model is greater than the likelihood of emission by the null model: $\alpha_{P_{END}}(n) > \alpha_{N_{END}}(n)$, where P and N refer to the positive and null models respectively and the sentence has n segments.
2. In the Viterbi path for the positive model, there are segments aligned with states corresponding to *all* the domains of the relation. For example, for the *subcellular-localization* relation, the Viterbi path for a sentence must pass through a state with the *PROTEIN* label and a state with the *LOCATION* label.

Note that even after phrases have been identified in this way, the extraction task is not quite complete, since some of the phrases might contain words other than those that belong in an extracted tuple. Consider the example in Figure 2a. The *LOCATION* phrase contains the word “the” in addition to the location. Therefore, tuple extraction with these models must include a post-processing phase in which such extraneous words are stripped away before tuples are returned. We do not address this issue here. Instead, we consider a prediction to be correct if the model correctly identifies the phrases containing the target tuple as a subphrase.

It is possible to have multiple predicted segments for each domain of the relation. In this case, we must decide which combinations of segments constitute tuples. We do this using two simple rules:

1. Associate segments in the order in which they occur. Thus for *subcellular-localization*, the first segment matching a *PROTEIN* state is associated with the first segment matching a *LOCATION* state, and so on.
2. If there are fewer segments containing an element of some domain, use the last match of this domain to construct the remaining tuples. For instance, if we predicted one *PROTEIN* phrase P_1 and two *LOCATION* phrases L_1 and L_2 , we would create two tuples based on $\langle P_1, L_1 \rangle$ and $\langle P_1, L_2 \rangle$.

3.1 Experiments

In the experiments presented in this section, we test our hypothesis that incorporating phrase-level sentence structure into our model provides improved extraction performance in terms of precision and recall. We test this hypothesis by comparing against several hidden Markov models that represent less information about the grammatical structure of sentences. Henceforth, we refer to the model described above as the *Phrase Model*.

The first model we compare against, which we call the *POS Model*, is based on the representation shown in Figure 2b. This model represents some grammatical information, in that it associates a type with each token indicating the part-of-speech (POS) tag for the word (as determined by Sundance). However, unlike the *Phrase Model*, the *POS model* represents sentences as sequences of tokens, not phrases. This model is comparable in size to the *Phrase Model*. The positive component of this model has 17 states without labels and six to ten states with labels (depending on the training set). The null component of the model has 17 states without labels.

The other models we consider, which we call the *Token Models*, are based on the representation shown in Figure 2c. This representation treats a sentence simply as a sequence of words. We investigate two variants that employ this representation. The simpler of the two hidden Markov models based on this representation, which we refer to as *Token Model 1*, has three states in its positive model and one state in its null model (not counting the **START** and **END** states). None of the states in this model have types. Two of the states in the positive model represent the domains of the binary target relation, while the remaining states have no labels. The role of the latter set of states is to model all tokens that do not correspond to the domains of the target relation. A more complex version of this model, which is illustrated in Figure 4, has three unlabeled states in its positive model. We define the transitions and train these models in such a way that these three states can specialize to (i) tokens that come before any relation instances, (ii) tokens that are interspersed between the domains of relation instances, and (iii) tokens that come after relation instances.

The training algorithm used for the *POS Model* is identical to that used for the *Phrase Model*. The training algorithm for the *Token Models* is essentially the same, except that there are no type constraints on either the tokens or states.

Since we consider a prediction made by the *Phrase Model* to be correct if it simply identifies the phrases containing the words of the tuple, we use a similar criterion to decide if the predictions made by the *POS Model* and *Token Models* are

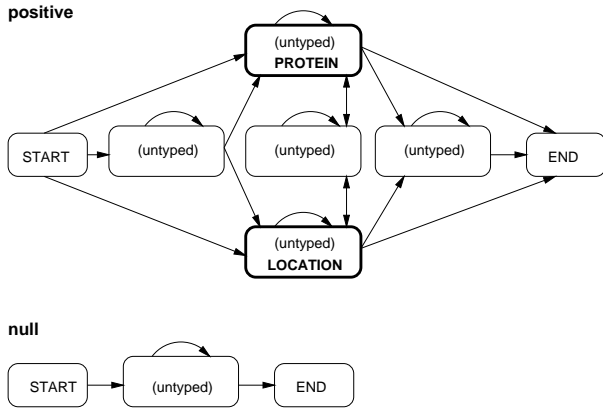


Figure 4: The architecture of *Token Model 2*.

correct. We consider *POS Model* and *Token Model* predictions to be correct if the labeled states of these models identify sequences of tokens that contain the words of the tuple. These models are not penalized for extracting extra adjacent words along with the actual words of a target tuple.

We process the HMM input data (after parsing in cases where Sundance is used) by stemming words with the Porter algorithm [Porter, 1980], and replacing words that occur only once in a training set with a generic UNKNOWN token. The statistics for this token are then used by the model while emitting out-of-vocabulary words encountered during prediction. Similarly, numbers are mapped to a generic NUMBER token.

Positive predictions are ranked by a confidence measure which is computed as the ratio of the likelihood of the Viterbi path of a sentence through a model to the likelihood of the model to emit that sentence, i.e. $\text{confidence}(s) = \delta_{\text{END}}(n) / \alpha_{\text{END}}(n)$. Here s is a sentence of n segments, $\delta_{\text{END}}(n)$ is the likelihood of the most probable path of all n segments threaded through to the END state, and $\alpha_{\text{END}}(n)$ is the comparable value calculated by the Forward algorithm. We construct precision-recall graphs for our models by varying a threshold on the confidence measures.

For both data sets we measure precision and recall using 5-fold cross validation. The data is partitioned such that all of the sentences from a given MEDLINE abstract are in the same fold. This procedure ensures that our experiments model the nature of the real application setting. For training, we sample the negative instances so that there are an equal number of positive and negative instances per fold. We have observed that we get better recall consistently by doing this.

Figure 5 shows the precision-recall curves for the subcellular-localization data set. The curve for the *Phrase Model* is superior to the curves for both *Token Models*. At low levels of recall, the *POS Model* exhibits slightly higher precision than the *Phrase Model*, but the latter is superior at higher recall levels, and the *Phrase Model* has a significantly higher recall endpoint. These results suggest that there is value in representing grammatical structure in the HMM architectures, but the *Phrase Model* is not definitively more accurate.

Figure 6 shows the precision-recall curves for the

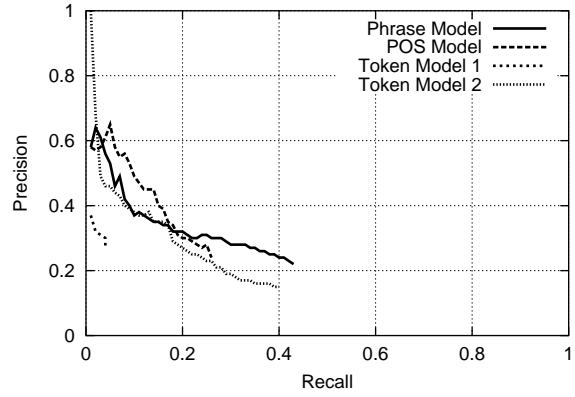


Figure 5: Precision vs. recall for the four models on the subcellular-localization data set.

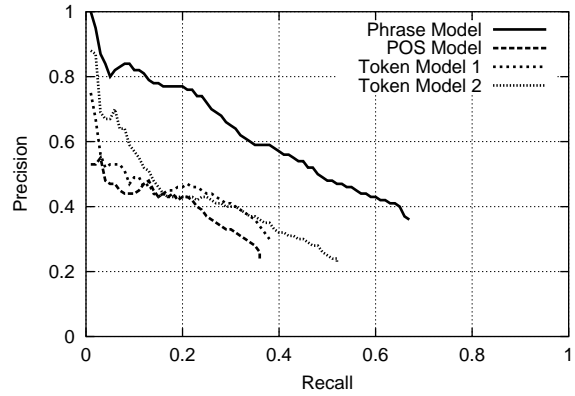


Figure 6: Precision vs. recall for the four models on the disorder-association data set.

disorder-association data set. Here, the differences are much more pronounced. The *Phrase Model* achieves significantly higher levels of precision than any of the other models, including the *POS Model*. The recall endpoint for the *Phrase Model* is also superior to those of the other models. We conclude that the experiments presented here support our hypothesis that incorporating sentence structure into the models we build results in better extraction accuracy.

4 Improving Parameter Estimates

Standard HMM training algorithms, like Baum-Welch, are designed to maximize the likelihood of the data given the model. Specifically, if s_i is a sentence in the training set, Baum-Welch (and the method we used earlier) tries to find parameters $\hat{\theta}$ such that

$$\hat{\theta} = \arg \max_{\theta} \prod_i \Pr(s_i | \theta).$$

We hypothesize that more accurate models can be learned by training with an objective function that aims to maximize the likelihood of predicting the correct sequence of *labels* for a given sentence (as before, we assume that states and phrases

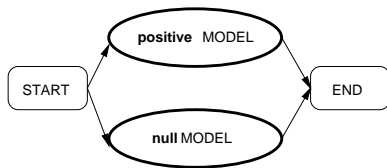


Figure 7: Combined model architecture. The *positive* and *null* models refer to the corresponding models in Figure 3.

without labels have an implicit *empty* label). Let c_i be the known sequence of labels for a sentence s_i in our training set. We would like to estimate parameters $\hat{\theta}$ such that

$$\hat{\theta} = \arg \max_{\theta} \prod_i \Pr(c_i | s_i, \theta) \quad (3)$$

$$= \arg \max_{\theta} \prod_i \frac{\Pr(c_i, s_i | \theta)}{\Pr(s_i | \theta)}. \quad (4)$$

This is similar to the task of optimizing parameters to recover the sequence of states given a set of observations [McCallum *et al.*, 2000]. Krogh [1994] has devised an HMM training algorithm that tries to optimize this criterion. After transforming this objective function into one which aims to minimize the negative log likelihood of the above equation, the following incremental update rule is obtained:

$$\theta_j^{new} = N(\theta_j^{old} + \eta(m_j^i - n_j^i)) \quad (5)$$

where θ_j is the j^{th} parameter, m_j^i is the expected number of times θ_j is used by the i^{th} sentence on *correct* paths through the model, n_j^i is the expected number of times θ_j is used by the i^{th} sentence on *all* paths through the model, N is a normalizing constant, and η is the learning rate. The n and m terms can be calculated using the Forward-Backward procedure. Note that the update rule represents an online training procedure.

In our previous experiments, we used a separate null model to represent negative instances. We would like to use Krogh's algorithm with this configuration to observe if it results in more accurate models. However, the null model as we have described it is a separate entity which is trained separately. With this architecture, Krogh's algorithm would be unable to correct false positives in the training set since doing so might require adjusting the parameters of the positive model in response to a negative instance. To remedy this problem, we propose an alternative to having a separate null model, which we refer to as a *Combined Model*. A *Combined Model* consists of two submodels sharing common START and END states. A schematic is shown in figure 7. The shared START and END states allow the training algorithm to update parameters in both parts of the model in response to a given training sentence.

4.1 Experiments

To evaluate this algorithm, we train the *Combined Model* configuration on the subcellular-localization and the disorder-association data sets. We compare these models against the

Phrase Model trained on the corresponding data sets in our previous experiments.

The methodology for this experiment is the same as before. Note that for the *Combined Model*, prediction is simpler than with a separate null model, since it suffices to consider the Viterbi path of a sentence through the model to extract tuples, if any. We do not train the *Combined Model* to convergence to avoid overfitting. Instead, we set the number of iterations for which to do gradient descent to a fixed constant value of 100.

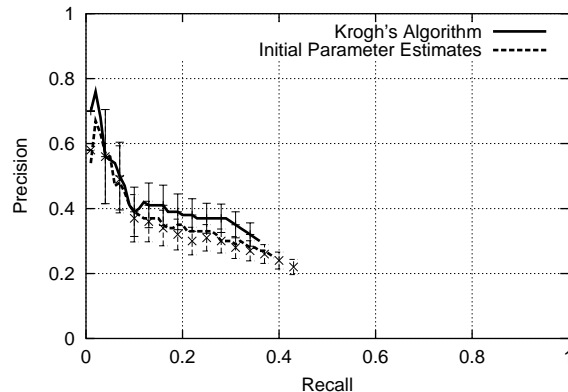


Figure 8: Effect of Krogh's Algorithm on the combined model for the subcellular-localization data set.

Figure 8 shows the precision-recall curves for this experiment for the subcellular-localization data set. For each precision-recall curve, we also show 95% confidence intervals. From the figure, we observe that there is some improvement in the precision of the model on this data set, while recall is held nearly constant. While the improvement is small, we have observed it consistently across the various model architectures we have explored. Figure 9 shows the corresponding precision-recall curves and confidence intervals for the experiment on the disorder-association data set. Here, the difference between the initial model and the trained model is more pronounced. The model trained with Krogh's algo-

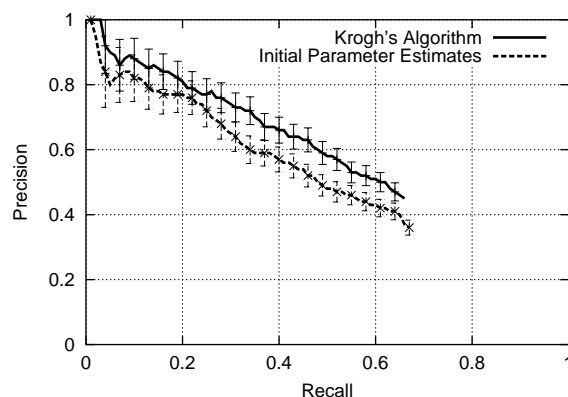


Figure 9: Effect of Krogh's Algorithm on the combined model for the disorder-association data set.

rithm has significantly better precision than the initial model, while maintaining a similar level of recall. We conclude that this training algorithm is appropriate for our task, and can improve accuracy, sometimes significantly.

5 Conclusion

We have presented two contributions to learning Hidden Markov Models for information extraction, and evaluated these contributions on two challenging biomedical domains. We have presented an approach to representing the grammatical structure of sentences in an HMM. Comparative experiments with other models lacking such information shows that this approach learns extractors that have increased precision and recall performance. We have also investigated the application of a training algorithm developed by Krogh to our models. This algorithm consistently provides an accuracy gain over our original models. We believe that these are promising approaches to the task of deriving information extractors for free text domains.

Acknowledgments

This research was supported in part by NIH Grant 1R01 LM07050-01, and NSF CAREER award IIS-0093016. The authors would like to thank Michael Waddell for his work on building the disorder-association data set, and Peter Andreae, Joseph Bockhorst, Tina Eliassi-Rad, and Jude Shavlik for critiquing the initial draft.

References

- [Baum, 1972] L. E. Baum. An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.
- [Center for Medical Genetics, 2001] Center for Medical Genetics, Johns Hopkins University and National Center for Biotechnology Information. Online Mendelian inheritance in man, OMIM (TM), 2001. <http://www.ncbi.nlm.nih.gov/omim/>.
- [Cestnik, 1990] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–150, Stockholm, Sweden, 1990. Pitman.
- [Freitag and McCallum, 1999] D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In *Working Notes of the AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, FL, 1999. AAAI Press.
- [Freitag and McCallum, 2000] D. Freitag and A. McCallum. Information extraction with HMM structures learned by stochastic optimization. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX, 2000. AAAI Press.
- [Hodges *et al.*, 1998] P. E. Hodges, W. E. Payne, and J. I. Garrels. Yeast protein database (YPD): A database for the complete proteome of *saccharomyces cerevisiae*. *Nucleic Acids Research*, 26:68–72, 1998.
- [Krogh, 1994] A. Krogh. Hidden Markov models for labeled sequences. In *Proceedings of the Twelfth International Conference on Pattern Recognition*, pages 140–144, Jerusalem, Israel, 1994. IEEE Computer Society Press.
- [Leek, 1997] T. Leek. Information extraction using hidden Markov models. Master's thesis, Department of Computer Science and Engineering, University of California, San Diego, CA, 1997.
- [McCallum *et al.*, 2000] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, Stanford, CA, 2000. Morgan Kaufmann.
- [National Library of Medicine, 2001] National Library of Medicine. The MEDLINE database, 2001. <http://www.ncbi.nlm.nih.gov/PubMed/>.
- [Porter, 1980] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):127–130, 1980.
- [Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [Riloff, 1996] E. Riloff. An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, 85:101–134, 1996.
- [Riloff, 1998] E. Riloff. The Sundance sentence analyzer, 1998. <http://www.cs.utah.edu/projects/nlp/>.
- [Seymore *et al.*, 1999] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *Working Notes of the AAAI Workshop on Machine Learning for Information Extraction*, pages 37–42. AAAI Press, 1999.

Sequentially finding the N -Best List in Hidden Markov Models

Dennis Nilsson
Aalborg University
Bajers Vej 7 E, 9220 Aalborg Øst
Denmark
nilsson@math.auc.dk

Jacob Goldberger
The Weizmann Institute of Science
Rehovot, 76100
Israel
jacob@wisdom.weizmann.ac.il

Abstract

We propose a novel method to obtain the N -best list of hypotheses in hidden Markov model (HMM). We show that the entire information needed to compute the N -best list from the HMM trellis graph is encapsulated in entities that can be computed in a single forward-backward iteration that usually yields the most likely state sequence. The hypotheses list can then be extracted in a sequential manner from these entities without the need to refer back to the original data of the HMM. Furthermore, our approach can yield significant savings of computational time when compared to traditional methods.

1 Introduction

In many tasks of large vocabulary speech recognition it is desirable to find from the HMM graph the N most likely state sequences given the observed acoustic data. The recognizer chooses the utterance hypotheses on the basis of acoustic information and a relatively simple language model. The existence of an N -best list enable us to combine additional knowledge sources such as complicated acoustic and language models into the recognition process [Ostendorf 1991]. Given the additional knowledge sources the list of sentence can be rescored and reordered. Even without additional knowledge sources the N -best paradigm can be used to improve the recognition rate [Stolcke *et al.* 1997]. In this paper we concentrate on the step of computing the N -best list. The most likely state sequence can be found using a single iteration of the Viterbi algorithm [Rabiner 1989]. A direct generalization of this algorithm can be used to obtain the N best state sequences. The only change is that for each time index t and for each state we have to keep the N best subsequences terminating at this state. However in this generalized Viterbi approach we have to decide in advance on the size of N and we can not change it in the middle of the process. Several modification of this algorithm have been proposed in the last decade. These algorithms are based either on a Viterbi search of a trellis or on A^* search [Schwartz and Chow 1991] [Schwartz *et al.* 1996].

We propose a novel method to obtain the N -best list in HMM. The obtained algorithm is inspired by the divide and conquer algorithm in [Nilsson (1998)] for finding the M most

likely configurations in probabilistic expert systems. The present algorithm also has the advantage of being an anytime algorithm since we need not in forehand specify the number of N best hypothesis that is wanted. Furthermore, our algorithm can yield significant savings in computational time compared to the traditional Viterbi algorithm.

2 Basic Structure

Consider a HMM with m hidden Markovian random variables X_1, \dots, X_m and m observed variables Y_1, \dots, Y_m such that the distribution of Y_t is determined by X_t . Denote $X = \{X_1, \dots, X_m\}$ and $Y = \{Y_1, \dots, Y_m\}$. Typical values that X and Y can take are denoted $x = (x_1, \dots, x_m)$ and $y = (y_1, \dots, y_m)$ respectively. The joint probability function is:

$$P(x, y) = P(x_1) \prod_{t=2}^m P(x_t | x_{t-1}) \prod_{t=1}^m P(y_t | x_t) \quad (1)$$

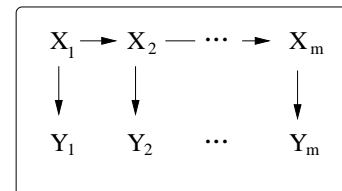


Figure 1: The HMM structure

This paper deals with the following decoding problem: Given observations $y_1 \dots y_m$, find the N most likely state sequences of the unobserved state-variables. In other words we want to find the N values of x that maximize the conditional probability function $P(x, y)$ (viewed as a function of x). A single iteration of the forward-backward algorithm [Rabiner 1989] yields the following terms for each time index t :

$$\begin{aligned} f_t(s) &= \max_{\{x|x_t=s\}} P(x, y) \\ f_{t,t+1}(s, s') &= \max_{\{x|(x_t, x_{t+1})=(s, s')\}} P(x, y) \end{aligned} \quad (2)$$

We shall show that the entire information needed to compute the N -best list is encapsulated in the expressions defined in

(2). In other words, once $f_t(s)$ and $f_{t,t+1}(s, s')$ are given, there is no need to refer again to the trellis graph.

As a first example of the usefulness of f_t and $f_{t,t+1}$, we apply them to obtain the most likely state sequence. One can observe that the probability of the most likely state sequence is :

$$\max_x P(x, y) = \max_s f_t(s)$$

This equality remains the same for each index $t = 1, \dots, m$. To find the most likely state sequence $\hat{x} = (\hat{x}_1, \dots, \hat{x}_m)$, when this is uniquely determined, we note that

$$\hat{x}_t = \arg \max_s f_t(s), \quad t = 1, \dots, m$$

However to allow for the possibility that the maximizing state sequence is not unique, it is better to apply the following routine:

$$\begin{aligned} (\hat{x}_1, \hat{x}_2) &= \arg \max_{(s, s')} f_{1,2}(s, s') \\ \hat{x}_t &= \arg \max_s f_{t-1,t}(\hat{x}_{t-1}, s), \quad t > 2 \end{aligned} \quad (3)$$

3 Introducing the algorithm

Denote the entire ensemble of possible state sequences by \mathcal{X} and let the L th most likely state sequence be denoted

$$x^L = (x_1^L, \dots, x_m^L).$$

Suppose at some stage in the algorithm that x^1, \dots, x^{L-1} have been identified. Then x^L is found by performing the following steps:

PARTITION PHASE: Here $\mathcal{X} \setminus \{x^1, \dots, x^{L-1}\}$ is partitioned into subsets.

CANDIDATE PHASE: For each subset in the above partitioning we compute the probability of its most likely state sequence. These probabilities are referred to as the ‘candidates’.

IDENTIFICATION PHASE: The state sequence associated with the highest candidate is identified.

In **PARTITION PHASE**, there is a large number of possible partitions that one may consider. We seek a partitioning with two properties. Firstly, it must be easy to compute the candidates. Secondly, the number of subsets in the partitioning must be small, since eventually we need to compare the different candidates.

In the successively subsections it is shown how the second and third most likely state sequences are found. In Section 4 the general case is considered.

3.1 The second most likely state sequence

Suppose the most likely state sequence x^1 has now been identified. To identify x^2 we carry out the following steps.

PARTITION PHASE

Here, $\mathcal{X} \setminus \{x^1\}$ is partitioned into subsets A_1, \dots, A_m defined by

$$\begin{aligned} A_1 &= \{x \mid x_1 \neq x_1^1\} \\ A_i &= \{x \mid x_1 = x_1^1, \dots, x_{i-1} = x_{i-1}^1, x_i \neq x_i^1\}, \quad i \geq 2. \end{aligned} \quad (4)$$

CANDIDATE PHASE

We let $\hat{P}(A_i)$ stand for the probability of the most likely state sequence in the subset A_i , i.e.

$$\hat{P}(A_i) = \max_{x \in A_i} P(x, y),$$

and use a similar notation for other subsets.

The functions $f_{t,t+1}$ satisfy that

$$\begin{aligned} \hat{P}(A_1) &= \max_{\{(s, s') \mid s \neq x_1^1\}} f_{1,2}(s, s') \\ \hat{P}(A_i) &= \max_{s \neq x_i^1} f_{i-1,i}(x_{i-1}^1, s), \quad i > 2 \end{aligned} \quad (5)$$

So, the second most likely state sequence has probability

$$P(x^2, y) = \max_i \hat{P}(A_i).$$

IDENTIFICATION PHASE

If $\hat{P}(A_i) = \max\{\hat{P}(A_j) : j = 1, \dots, m\}$, then $x^2 \in A_i$, and x^2 can be identified by carrying out the following steps:

- $x_j^2 = x_j^1$ for $j = 1, \dots, i-1$.
- $x_i^2 = \arg \max_{s \neq x_i^1} f_{i-1,i}(x_{i-1}^1, s)$
- $x_k^2 = \arg \max_s f_{k-1,k}(x_{k-1}^2, s)$ for $k = i+1, \dots, m$.

3.2 The third most likely state sequence

The identification of the third most likely state sequence is done by a similar procedure as the case with x^2 .

PARTITION PHASE

Here, a partition of $\mathcal{X} \setminus \{x^1, x^2\}$ is constructed by refining the above partitioning of $\mathcal{X} \setminus \{x^1\}$. This is done in the following way.

Suppose $x^2 \in A_i$. Then we partitioning $A_i \setminus \{x^2\}$ into subsets B_i, \dots, B_m defined by

$$\begin{aligned} B_i &= \{x \mid x_1 = x_1^1, \dots, x_{i-1} = x_{i-1}^1, x_i \notin \{x_i^1, x_i^2\}\} \\ B_k &= \{x \mid x_1 = x_1^1, \dots, x_{k-1} = x_{k-1}^2, x_k \neq x_k^2\}, \quad k > i. \end{aligned} \quad (6)$$

Thus, the subsets in $\{B_j : j \geq i\}$ and $\{A_j : j \neq i\}$ constitute a partitioning of $\mathcal{X} \setminus \{x^1, x^2\}$.

CANDIDATE PHASE

As we shall prove in the next section (Theorem 1), the probability $\hat{P}(B_k)$ for $k \geq i$ is a simple function of $\hat{P}(A_i)$:

$$\begin{aligned} \hat{P}(B_i) &= \hat{P}(A_i) \frac{\max_{s \notin \{x_i^1, x_i^2\}} f_{i-1,i}(x_{i-1}^1, s)}{f_{i-1,i}(x_{i-1}^1, x_i^2)} \\ \hat{P}(B_k) &= \hat{P}(A_i) \frac{\max_{s \neq x_k^2} f_{k-1,k}(x_{k-1}^2, s)}{f_{k-1,k}(x_{k-1}^2, x_k^2)}, \quad k > i. \end{aligned} \quad (7)$$

Now, the third most likely state sequence has probability

$$P(x^3, y) = \max \left\{ \max_{j: j \neq i} \hat{P}(A_j), \max_{j: j \geq i} \hat{P}(B_j) \right\}.$$

IDENTIFICATION PHASE

If the third most likely state sequence x^3 belongs to one of the subsets A_j ($j \neq i$), then it can be identified in a similar way as x^2 was identified. On the other hand, if $x^3 \in B_j$ for some $j \geq i$, then

$$(x_1^3, \dots, x_{j-1}^3) = (x_1^2, \dots, x_{j-1}^2),$$

and the sequence x_j^3, \dots, x_m^3 is now successively identified as follows:

$$x_j^3 = \begin{cases} \arg \max_{s \notin \{x_j^1, x_j^2\}} f_{j-1,j}(x_{j-1}^3, s) & \text{if } j = i \\ \arg \max_{s \neq x_j^2} f_{j-1,j}(x_{j-1}^3, s) & \text{if } j > i \end{cases}$$

$$x_k^3 = \arg \max_s f_{k-1,k}(x_{k-1}^3, s), \text{ for } k = j+1, \dots, m$$

Example To illustrate our algorithm, we consider a simple HMM with 10 hidden variables X_1, \dots, X_{10} , and 10 observed variables Y_1, \dots, Y_{10} . Each of the variables are assumed binary with possible states y and n . The initial conditional probability functions associated with the HMM are given in Table 1.

$P(x_1)$	<table><tr><th>n</th><th>y</th></tr><tr><td>.6</td><td>.4</td></tr></table>	n	y	.6	.4												
n	y																
.6	.4																
$P(x_t x_{t-1})$	<table><tr><th>x_t</th><th>x_{t-1}</th></tr><tr><td><table><tr><th>n</th><th>y</th></tr><tr><td>.6</td><td>.4</td></tr><tr><td>.2</td><td>.8</td></tr></table></td><td><table><tr><th>n</th><th>y</th></tr><tr><td>.9</td><td>.1</td></tr><tr><td>.3</td><td>.7</td></tr></table></td></tr></table>	x_t	x_{t-1}	<table><tr><th>n</th><th>y</th></tr><tr><td>.6</td><td>.4</td></tr><tr><td>.2</td><td>.8</td></tr></table>	n	y	.6	.4	.2	.8	<table><tr><th>n</th><th>y</th></tr><tr><td>.9</td><td>.1</td></tr><tr><td>.3</td><td>.7</td></tr></table>	n	y	.9	.1	.3	.7
x_t	x_{t-1}																
<table><tr><th>n</th><th>y</th></tr><tr><td>.6</td><td>.4</td></tr><tr><td>.2</td><td>.8</td></tr></table>	n	y	.6	.4	.2	.8	<table><tr><th>n</th><th>y</th></tr><tr><td>.9</td><td>.1</td></tr><tr><td>.3</td><td>.7</td></tr></table>	n	y	.9	.1	.3	.7				
n	y																
.6	.4																
.2	.8																
n	y																
.9	.1																
.3	.7																

Table 1: The initial probabilities in the HMM

Suppose we obtain ‘evidence’ of the form

$$y = (y_1, y_2, \dots, y_{10}) = (n, n, y, y, y, y, n, n, y, y).$$

To compute the N -best list given evidence, we initially compute the $f_{t,t+1}$ functions shown in Table 2. By applying the routine in (3), we can now obtain the most likely state sequence:

$$x^1 = (n, n, y, y, y, y, y, y, y, y)$$

with probability $P(x^1, y) = .2591 \cdot 10^{-3}$.

For the computation of the second most likely state sequence we proceed as follows. First, we partition the space $\mathcal{X} \setminus \{x^1\}$ into subsets A_1, \dots, A_{10} defined in (4):

$$\begin{aligned} A_1 &= \{x \mid x = (y, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot)\} \\ A_2 &= \{x \mid x = (n, y, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot)\} \\ A_3 &= \{x \mid x = (n, n, n, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot)\} \\ A_4 &= \{x \mid x = (n, n, y, n, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot)\} \\ A_5 &= \{x \mid x = (n, n, y, y, n, \cdot, \cdot, \cdot, \cdot, \cdot)\} \\ A_6 &= \{x \mid x = (n, n, y, y, y, n, \cdot, \cdot, \cdot, \cdot)\} \\ A_7 &= \{x \mid x = (n, n, y, y, y, y, n, \cdot, \cdot, \cdot)\} \\ A_8 &= \{x \mid x = (n, n, y, y, y, y, y, n, \cdot, \cdot)\} \\ A_9 &= \{x \mid x = (n, n, y, y, y, y, y, y, n, \cdot)\} \\ A_{10} &= \{x \mid x = (n, n, y, y, y, y, y, y, y, n)\} \end{aligned}$$

$f_{1,2}$	x_2	$f_{2,3}$	x_3												
x_1	n	x_2	n												
y	y	y	y												
	<table> <tr><th>n</th><th>y</th></tr> <tr><td>.2590</td><td>.1151</td></tr> <tr><td>.0192</td><td>.0512</td></tr> </table>	n	y	.2590	.1151	.0192	.0512		<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0278</td><td>.2590</td></tr> <tr><td>.0021</td><td>.1151</td></tr> </table>	n	y	.0278	.2590	.0021	.1151
n	y														
.2590	.1151														
.0192	.0512														
n	y														
.0278	.2590														
.0021	.1151														
$f_{3,4}$	x_4	$f_{4,5}$	x_5												
x_3	n	x_4	n												
y	y	y	y												
	<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0030</td><td>.0278</td></tr> <tr><td>.0046</td><td>.2590</td></tr> </table>	n	y	.0030	.0278	.0046	.2590		<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0005</td><td>.0046</td></tr> <tr><td>.0046</td><td>.2590</td></tr> </table>	n	y	.0005	.0046	.0046	.2590
n	y														
.0030	.0278														
.0046	.2590														
n	y														
.0005	.0046														
.0046	.2590														
$f_{5,6}$	x_6	$f_{6,7}$	x_7												
x_5	n	x_6	n												
y	y	y	y												
	<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0025</td><td>.0046</td></tr> <tr><td>.0234</td><td>.2590</td></tr> </table>	n	y	.0025	.0046	.0234	.2590		<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0234</td><td>.0046</td></tr> <tr><td>.2185</td><td>.2590</td></tr> </table>	n	y	.0234	.0046	.2185	.2590
n	y														
.0025	.0046														
.0234	.2590														
n	y														
.0234	.0046														
.2185	.2590														
$f_{7,8}$	x_8	$f_{8,9}$	x_9												
x_7	n	x_8	n												
y	y	y	y												
	<table> <tr><th>n</th><th>y</th></tr> <tr><td>.2185</td><td>.0971</td></tr> <tr><td>.0971</td><td>.2590</td></tr> </table>	n	y	.2185	.0971	.0971	.2590		<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0234</td><td>.2185</td></tr> <tr><td>.0046</td><td>.2590</td></tr> </table>	n	y	.0234	.2185	.0046	.2590
n	y														
.2185	.0971														
.0971	.2590														
n	y														
.0234	.2185														
.0046	.2590														
$f_{9,10}$	x_{10}														
x_9	n														
y	y														
	<table> <tr><th>n</th><th>y</th></tr> <tr><td>.0050</td><td>.0234</td></tr> <tr><td>.0093</td><td>.2590</td></tr> </table>	n	y	.0050	.0234	.0093	.2590								
n	y														
.0050	.0234														
.0093	.2590														

Table 2: The $f_{t,t+1}$ functions (the numbers are multiplied by 10^3)

Next, we use (5) to compute the probability of the most likely state sequence in each subsets A_i :

$$\begin{aligned} \hat{P}(A_1) &= \max\{f_{1,2}(y, n), f_{1,2}(y, y)\} = .0512 \\ \hat{P}(A_2) &= f_{1,2}(n, y) = .1151 \\ \hat{P}(A_3) &= f_{2,3}(n, n) = .0278 \\ \hat{P}(A_4) &= f_{3,4}(y, n) = .0046 \\ \hat{P}(A_5) &= f_{4,5}(y, n) = .0046 \\ \hat{P}(A_6) &= f_{5,6}(y, n) = .0234 \\ \hat{P}(A_7) &= f_{6,7}(y, n) = .2185 \\ \hat{P}(A_8) &= f_{7,8}(y, n) = .0971 \\ \hat{P}(A_9) &= f_{8,9}(y, n) = .0046 \\ \hat{P}(A_{10}) &= f_{9,10}(y, n) = .0093 \end{aligned}$$

Thus, the second most like state sequence x^2 belongs to subset A_7 , and can be found by carrying out steps (see Section 3.1):

$$(x_1^2, x_2^2, x_3^2, x_4^2, x_5^2, x_6^2, x_7^2) = (n, n, y, y, y, y, n),$$

and

$$\begin{aligned} x_8^2 &= \arg \max_s \{f_{7,8}(n, s)\} = n \\ x_9^2 &= \arg \max_s \{f_{8,9}(n, s)\} = y \\ x_{10}^2 &= \arg \max_s \{f_{9,10}(y, s)\} = y \end{aligned}$$

This identifies the second most likely state sequence.

To compute the third most likely state sequence we proceed in a similar manner: First, we partition $A_7 \setminus \{x^2\}$ in subsets of the form in (6):

$$\begin{aligned}
B_7 &= \{x \mid x = (n, n, y, y, y, y, \cdot, \cdot, \cdot) \wedge x_7 \notin \{n, y\}\} \\
B_8 &= \{x \mid x = (n, n, y, y, y, y, n, y, \cdot, \cdot)\} \\
B_9 &= \{x \mid x = (n, n, y, y, y, y, n, n, n, \cdot)\} \\
B_{10} &= \{x \mid x = (n, n, y, y, y, y, n, n, y, n)\}
\end{aligned}$$

Then, we compute the probability of the most likely state sequence within each of the subsets B_i . From (7) we immediately obtain ($\hat{P}(B_7) = 0$ since $B_7 = \emptyset$):

$$\begin{aligned}
\hat{P}(B_8) &= \hat{P}(A_7) \frac{f_{7,8}(n,y)}{f_{7,8}(n,n)} = .2185 \frac{.0971}{.2185} = .0971 \\
\hat{P}(B_9) &= \hat{P}(A_7) \frac{f_{8,9}(n,n)}{f_{8,9}(n,y)} = .2185 \frac{.0234}{.2185} = .00234 \\
\hat{P}(B_{10}) &= \hat{P}(A_7) \frac{f_{9,10}(y,n)}{f_{9,10}(y,y)} = .2185 \frac{.0093}{.2591} = .0078
\end{aligned}$$

Thus, the third most like state sequence x^3 belongs to subset A_2 , since

$$\max_{i \neq 7} \{\hat{P}(A_i), \hat{P}(B_8), \hat{P}(B_9), \hat{P}(B_{10})\} = \hat{P}(A_2),$$

and can be found by carrying out the steps described in Section 3.2. This is left to the reader.

Table 3 shows the three most likely state sequences.

	Configuration	Probability (multiplied by 10^3)
x^1	$(n, n, y, y, y, y, y, y, y, y)$	$P(x^1, y) = .2591$
x^2	$(n, n, y, y, y, y, n, n, y, y)$	$P(x^2, y) = .2185$
x^3	$(n, y, y, y, y, y, n, n, y, y)$	$P(x^3, y) = .1151$

Table 3: The 3-best list

4 The general algorithm

The general algorithm for computing the N -best list identifies the N most likely state sequences in a sequentially manner.

Let \mathcal{X}_i denote the possible states that the variable X_i can take.

Suppose at some stage in the algorithm that

- x^1, \dots, x^L have been identified;
- A partitioning, say \mathcal{P}^L , of $\mathcal{X} \setminus \{x^1, \dots, x^{L-1}\}$ is given.
- For each element D in \mathcal{P}^L , $\hat{P}(D)$ is known.
- $x^L \in D$ for some element D in the partitioning \mathcal{P}^L , and D has the following form:

$$D = \{x \mid x_1 = x'_1, \dots, x_{i-1} = x'_{i-1}, x_i \notin \mathcal{X}'_i\}, \quad (8)$$

where x'_1, \dots, x'_{i-1} are known states, and \mathcal{X}'_i is a known subset of \mathcal{X}_i .

The tasks involved in proving the correctness of our algorithm consist of:

Partition-phase: Construct a partitioning of $D \setminus \{x^L\}$;

Candidate-phase: Compute the probability of the most likely state-sequence in each element in the partitioning of $D \setminus \{x^L\}$;

Identification-phase:

- Compare the ‘new candidates’ computed above with the remaining candidates.
- Identify x^{L+1} .

These three tasks are described in subsequent subsections.

4.1 Partition phase

Suppose $x^L \in D$, where D has the form in (8). For the following analysis it is convenient to write D as

$$D = \{x \mid x_1 = x_1^L, \dots, x_{i-1} = x_{i-1}^L, x_i \notin \mathcal{X}'_i\}. \quad (9)$$

For $k = i, \dots, m$ we define the subsets D_k as

$$D_k = \{x \mid x_1 = x_1^L, \dots, x_{k-1} = x_{k-1}^L, x_k \notin \bar{\mathcal{X}}_k\}, \quad (10)$$

where $\bar{\mathcal{X}}_k$ is a subset of \mathcal{X}_k given by

$$\bar{\mathcal{X}}_k = \begin{cases} \mathcal{X}'_k \cup \{x_k^L\} & \text{if } k = i \\ \{x_k^L\} & \text{if } k > i \end{cases}$$

The reader may easily verify that the subsets D_i, \dots, D_m partition $D \setminus \{x^L\}$. Furthermore, it can be seen that each element D_k has a similar form as that of D .

4.2 Candidate phase

In this section we provide an efficient method to compute the probabilities $\hat{P}(D_k)$ ($k = i, \dots, m$). The method presented here is similar to the the method in [Nilsson 1998], in that it locally computes the probabilities of all candidates directly from the functions $f_{t,t+1}$.

A keypoint in proving our main result is that we can write the joint probability function expressed in (1) as

$$P(x, y) = \frac{\prod_{t=1}^{m-1} f_{t,t+1}(x_t, x_{t+1})}{\prod_{t=2}^{m-1} f_t(x_t)}. \quad (11)$$

For a proof, the interested reader is referred to [Dawid 1992], where it is shown for more general graphical models, the junction trees.

In addition, we will use that the functions $f_{t,t+1}$ have the following property, termed *max-consistency*:

$$\max_{x_{t+1}} f_{t,t+1}(x_t, x_{t+1}) = f_t(x_t). \quad (12)$$

Now we have

Lemma 1 *Let E be a subset given by*

$$E = \{x : x_1 = x_1^*, \dots, x_i = x_i^*\}.$$

Then

$$\hat{P}(E) = \max_{x \in E} P(x, y) = \frac{\prod_{t=1}^{i-1} f_{t,t+1}(x_t^*, x_{t+1}^*)}{\prod_{t=2}^{i-1} f_t(x_t^*)}.$$

Proof: The probability $\hat{P}(E)$ can be found by instantiating $x_1 = x_1^*, \dots, x_i = x_i^*$ in (11), and then maximize over the remaining variables. Because of max-consistency (12) this reduces to

$$\max_{x \in E} P(x, y) = \frac{\prod_{t=1}^{i-1} f_{t,t+1}(x_t^*, x_{t+1}^*)}{\prod_{t=2}^{i-1} f_t(x_t^*)},$$

which completes the proof. •

Lemma 2 Let D be given as in (9), and suppose

$$x^L = \arg \max_{x \in D} P(x, y).$$

Then for all $k = i, \dots, m$ we have

$$\hat{P}(D) = \max_{x \in D} P(x, y) = \frac{\prod_{t=1}^{k-1} f_{t,t+1}(x_t^L, x_{t+1}^L)}{\prod_{t=2}^{k-1} f_t(x_t^L)}.$$

Proof: Let $k \geq i$, and define D' as

$$D' = \{x : x_1 = x_1^L, \dots, x_k^L\}.$$

By Lemma 1 it suffices to prove that

$$\max_{x \in D'} P(x, y) = \max_{x \in D} P(x, y).$$

Clearly,

$$\max_{x \in D'} P(x, y) \leq \max_{x \in D} P(x, y)$$

since $D' \subseteq D$. Furthermore, we have that

$$\max_{x \in D'} P(x, y) \geq \max_{x \in D} P(x, y)$$

because $x^L \in D'$ and $x^L = \arg \max_{x \in D} P(x, y)$. The result follows. •

Now we are ready to state the main theorem. It presents a straightforward procedure for finding the probability of each subset D_k (defined in (10)) from the probability $\hat{P}(D)$.

Theorem 1 Let D be given as in (9), and suppose

$$x^L = \arg \max_{x \in D} P(x, y).$$

Then for all D_k defined in (10) we have:

$$\hat{P}(D_k) = \hat{P}(D) \frac{\max_{s \notin \bar{\mathcal{X}}_k} f_{k,k+1}(x_k^L, s)}{f_{k,k+1}(x_k^L, x_{k+1}^L)}.$$

Proof: By Lemma 2 we have

$$\hat{P}(D) = \frac{\prod_{t=1}^{k-1} f_{t,t+1}(x_t^L, x_{t+1}^L)}{\prod_{t=2}^{k-1} f_t(x_t^L)}, \quad (13)$$

Furthermore, since D_k can be written as

$$D_k = \cup_{s: s \notin \bar{\mathcal{X}}_k} \{x : x_1 = x_1^L, \dots, x_{k-1} = x_{k-1}^L, x_k = s\}.$$

we have from Lemma 1 that

$$\hat{P}(D_k) = \frac{\prod_{t=1}^{k-2} f_{t,t+1}(x_t^L, x_{t+1}^L)}{\prod_{t=2}^{k-1} f_t(x_t^L)} \max_{s \notin \bar{\mathcal{X}}_k} f_{k-1,k}(x_{k-1}^L, s) \quad (14)$$

The result now follows from (13) and (14). •

4.3 Identification phase

Suppose that we are given a subset D of the form

$$D = \{x \mid x_1 = x_1', \dots, x_{i-1} = x_{i-1}', x_i \notin \mathcal{X}_i'\}.$$

and want to identify the most likely state sequence, say x^* , in D . First, one note that

$$x_j^* = x_j' \text{ for } j = 1, \dots, i-1.$$

Now the sequence x_i^*, \dots, x_m^* is successively found by

$$x_i^* = \arg \max_{s \notin \mathcal{X}_i'} f_{i-1,i}(x_{i-1}^*, s)$$

$$x_k^* = \arg \max_s f_{k-1,k}(x_{k-1}^*, s), \quad k = i+1, \dots, m.$$

4.4 The algorithm

A pseudo code for our N -best list algorithm is given below.

Procedure N -best list:

Step 1 (Identify x^1):

1. Compute the functions $f_{t,t+1}$;
2. Identify the most likely state sequences x^1 as shown in Section 3.1;

Step 2 (Identify x^2):

•PARTITION PHASE Partition $\mathcal{X} \setminus \{x^1\}$ in subsets A_1, \dots, A_m as in (4);

•CANDIDATE PHASE Compute $\hat{P}(A_i)$ as in Section 3.1.

•IDENTIFICATION PHASE

1. Set the Candidate List $CL = \{\hat{P}(A_i)\}$;
2. pick the highest candidate, say $\hat{P}(A_i)$, in CL ;
3. identify $x^2 \in A_i$ as in Section 3.1; .

Step 3 (Identify x^3, \dots, x^N)

For $L = 2, \dots, N-1$ do

•PARTITION PHASE:

1. Suppose $x^L \in D$;
2. Partition $D \setminus \{x^L\}$ in subsets D_k, \dots, D_m as in (10).

•CANDIDATE PHASE: Compute $\hat{P}(D_k), \dots, \hat{P}(D_m)$ as in Theorem 1.

•IDENTIFICATION PHASE:

1. Augment the Candidate List with the new candidates:

$$CL := CL \cup \{\hat{P}(D_k), \dots, \hat{P}(D_m)\};$$

2. pick the highest candidate, say $\hat{P}(D^*)$, from CL ;
3. identify $x^{L+1} = \arg \max_{x \in D^*} P(x, y)$ as in Section 4.3;
4. retract $\hat{P}(D^*)$ from the Candidate List: $CL := CL \setminus \{\hat{P}(D^*)\}$.

END

Accordingly, the probability of the L th most likely state sequence is either one of the candidates computed in the L th step of the algorithm or it is one of the candidates computed earlier and still on the Candidate List. The search over all the candidates can be effectively performed in the following way. We can sort the candidates on the Candidate List with their associated subsets. In PARTITION PHASE we split the subsets that contained x^{L-1} (and hence was at the top of the list) into several subsets. In CANDIDATE PHASE we merge the new candidates into the sorted list according to their probabilities. Now the state sequence x^L belongs to the subset at the top of the updated list.

Because the algorithm finds the N -best list sequentially, we need not in forehand specify the number of N best hypothesis that is wanted. In stead, we may chose to find the N most likely hypothesis whose total probability is at least α , where $0 < \alpha \leq 1$.

5 Complexity issues

We would claim that our method can be implemented with lower complexity than traditional methods for finding the N -best list.

To support this, we would briefly discuss the computational complexity of our method by computing the number of elementary operations needed to perform the various tasks in the algorithm. Let

$$\gamma = \max_i |\mathcal{X}_i| = \text{the maximum number of elements that a variable can have.}$$

The operations performed by our algorithm can be divided into three parts

- 1 Computing the functions $f_{t,t+1}$ and f_t defined in (2).
and for each $L = 1, \dots, N$:
- 2 In CANDIDATE PHASE, to compute the candidates as in Theorem 1.
- 3 In IDENTIFICATION PHASE, to compare the candidates on the Candidate List, and identify x^L .

Finding the functions in part 1 can be done by less than

$$\gamma^2 m \text{ operations.}$$

In CANDIDATE PHASE, we generate for each step L at most m new candidates. The computation of a candidate takes place via Theorem 1, and can be done with γ computations, i.e. computing all candidates in step $L = 1, \dots, N$ demands at most

$$\gamma m N \text{ computations.}$$

In IDENTIFICATION PHASE, we compare, in step L at most mL elements (since at most m candidates are generated in each step). If we store the elements in a 2-3 search tree, then the comparisons needed for inserting m new elements and update the search tree is $O(m \log(mL))$. Doing this for all $L = 1, \dots, N$, the total number of comparisons needed is in the order of

$$O\left(\sum_{L=2}^N m \log(mL)\right) = O(mN \log(mN)).$$

Finally, in IDENTIFICATION PHASE, we identify x^L as described in Section 4.3, and the number of comparisons needed is no larger than $m\gamma$. Thus the total number of comparisons cannot exceed

$$\gamma m N.$$

Adding up these terms, we obtain that the whole process of finding x^1, \dots, x^N is in the order of

$$O(\gamma^2 m) + O(m\gamma N) + O(mN \log(mN)).$$

To compare, a straightforward implementation of the Viterbi search as described in [Forney (1973)] uses in the order of

$$O(\gamma^2 N m).$$

We conclude that for large γ , our method can yield significant savings of computational time when comparing to traditional Viterbi search.

6 Conclusion

We have presented in this paper a novel method to compute the N most likely state sequences in HMMs. The algorithm has two advantages compared to traditional methods. Firstly it can yield significant gain in computational time. Secondly, it is an anytime algorithm, and thus is also effective in cases where we do not know in advance how many solutions are needed. The main concept is to perform a small preprocessing computation and then we can produce the sequences in an incremental manner. We have concentrated in this paper on applications of the algorithm to speech recognition problems. The proposed algorithm, however, can be applied to many other sources of information that are organized in a hidden Markov model e.g. analysis of DNA sequences and real time robot navigation.

Acknowledgment

This research was supported by DINA (Danish Informatics Network in the Agricultural Sciences), funded by the Danish Research Councils through their PIFT programme.

References

- Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, **2**, 25–36.
- Forney, G. (1973). The viterbi algorithm. *Proc. IEEE*, **61**, 268–78.
- Nilsson, D. (1998). An efficient algorithm for finding the m most probable configurations in probabilistic expert systems. *Statistics and Computing*, **8**, 159–73.
- Ostendorf, M. e. a. (1991). Integration of diverse recognition methodologies through reevaluation of n -best sentence hypotheses. In *Proceedings, DARPA Speech and Natural language Processing Workshop*, pp. 83–7.
- Rabiner, L. (1989). A tutorial on hidden markov models and selected application in speech recognition. *Proceedings of the IEEE*, **37**, (2), 257–86.
- Schwartz, R. and Chow, Y. (1991). A comparison of several approximate algorithms for finding multiple (n -best) sentence hypotheses. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 701–4.
- Schwartz, R., Nguyen, L., and Makhoul, J. (1996). Multiple-pass search strategies. In *Automatic Speech and Speaker recognition*, pp. 429–56. Kluwer Academic Publishers.
- Stolcke, A., Konig, Y., and Weintraub, M. (1997). Explicit word error minimization in n -best list rescoring. *Proc. EUROSPEECH*, **1**, 163–6.

NLP-driven IR: Evaluating Performances over a Text Classification task

Roberto Basili and Alessandro Moschitti and Maria Teresa Pazienza

University of Rome Tor Vergata

Department of Computer Science, Systems and Production

00133 Roma (Italy)

{basili,moschitti,pazienza}@info.uniroma2.it

Abstract

Although several attempts have been made to introduce Natural Language Processing (NLP) techniques in Information Retrieval, most ones failed to prove their effectiveness in increasing performances. In this paper Text Classification (TC) has been taken as the IR task and the effect of linguistic capabilities of the underlying system have been studied. A novel model for TC, extending a well know statistical model (i.e. Rocchio's formula [Ittner *et al.*, 1995]) and applied to linguistic features has been defined and experimented. The proposed model represents an effective feature selection methodology. All the experiments result in a significant improvement with respect to other purely statistical methods (e.g. [Yang, 1999]), thus stressing the relevance of the available linguistic information. Moreover, the derived classifier reaches the performance (about 85%) of the best known models (i.e. Support Vector Machines (SVM) and K -Nearest Neighbour (KNN)) characterized by an higher computational complexity for training and processing.

1 Introduction

Although in literature poor evidence assessing the relevance of Natural Language Processing (NLP) in improving Information Retrieval (IR) has been derived, a shared belief exists that linguistic processing can capture critical semantic aspects of document content that simple word matching cannot do. It has been also stressed (e.g. [Grefenstette, 1997]), that vector space models are inadequate to deal with retrieval from Web via commonly available simple and short queries. Language processing enables to enrich the document representation with semantic structures although the nature and methods for doing this are still under debate. Which specific information (structures and dependencies) can be suitably derived and which combined use of linguistic processes and IR models are to be applied represent still open questions.

In order to derive more insight on the above issues, a systematic experimental framework has to be defined, where tasks and performance factors can be assessed and measured.

Among other IR tasks, Text Classification (TC) is a promising process for our objectives. It plays a major role in retrieval/filtering processes. Moreover, given the rich experimental evidence on well-assessed benchmarking collections, TC better supports a comparative evaluation of the impact of linguistic information with respect to approaches based on word matching.

The classification problem is traditionally described as follows: given a set of classes ($C = \{C_1, \dots, C_n\}$, i.e. topics/subtopics labels, e.g. "Politics"/"Foreign Politics") and an extensive collection of examples classified into these classes, often called *training set*, the classification problem is the derivation of a decision function f that maps documents ($d \in D$) into one or more classes, i.e. $f : D \rightarrow 2^C$. As the specific topics (classes) are fixed, the extraction of *relevant* content from document (for retrieval purposes) can be more systematic (given their focused semantics) and less complex than in other IR scenarios. Therefore *TC* represents a suitable environment for testing the capabilities of *NLP* to capture such semantic aspects.

The role of linguistic content in *TC* relates to the definition of *features* able to provide specific and selective information about training and test documents and (consequently) about the target classes. Basic language processing capabilities allow to extend the knowledge on the words occurring in documents, e.g. their canonical form (i.e. the morphological derivation from a lemma) and their syntactic role (i.e. their part-of-speech (POS) in the input context). Previous works on NLP-driven text classification (e.g. [Basili *et al.*, 2000b]) suggest that such information improves performances. In particular, lemmatization and recognition (i.e. removal of Proper Nouns from the set of selective feature) provide a linguistically principled way to compress the features set (usually obtained by traditional crude methods like stop lists or statistical thresholds, e.g. χ^2). Statistical unsupervised terminological extraction has been also applied to TC training. It allows detecting more complex and relevant features, i.e. complex nominal groups typical of the different topics. The results are improved *TC* performances, although the contribution given by such modules has not yet been accurately measured.

The main reason for poor improvements (if any) when NLP is applied to IR is the noise introduced by the linguistic recognition errors which provides drawbacks comparable to the significant advantages. In the specific case of TC, when more

complex features (e.g. words and their POS tag or terminological units) are captured it can be even more difficult to select the relevant ones among the set of all features. Data sparseness effects (e.g. the lower frequency of n -grams wrt simple words) interact with wrong recognitions (e.g. errors in POS assignment) and the overall information about a class loses its potential selectivity.

The traditional solution is usually the *feature selection*, discussed for example in [Yang and Pedersen, 1997]. By applying statistical methods, (information gain, χ^2 , mutual information ...), the not relevant features are removed. Major drawbacks are that features irrelevant for a class may be removed even if they are important for another one. *Important* but rare or specific *features* may be cut in this way, as also noted in [Joachims, 1998]. The crucial issue here is how to give the right weight to a given feature in different classes. This is even more important when NLP (and, especially, terminology recognition) is applied: some technical terms can be perfectly valid features for a class and, at the same time, totally irrelevant or misleading for others.

In this paper an original TC model for selection and weighting of linguistically motivated features, as an extension of the the Rocchio classifier ([Ittner *et al.*, 1995]), has been designed and implemented. It has been experimented on feature sets extracted by NLP techniques: terminological expressions, part-of-speech tagged lemmas and proper nouns.

In Section 2 the novel feature selection model with its weighting capabilities is presented. Section 3 describes the NLP functionalities adopted for extracting the feature sets from the target documents during training and testing. In Section 4 the experiments aiming to measure the impact of the feature selection on the classification performances as well as of the contribution of linguistic information are described.

2 Text Classification

Two main approaches to the construction of a non-parametric classifier have been proposed and experimented in literature [Lewis *et al.*, 1996].

Profile-based (or linear) classifiers are characterized by a function f that is based on a similarity measure between the representation of the incoming document d and each class C_i . Both representations are vectors and similarity is traditionally estimated as the cosine angle between the two vectors. The description \vec{C}_i of each target class (C_i) is usually called *profile*, that is the vector summarizing the content of all the training documents pre-categorized under C_i . The vector components are called *features* and refer to independent dimensions in the space in which similarity is estimated. Traditional techniques (e.g. [Salton and Buckley, 1988; Salton, 1991]) make use of single words w as basic features. The i -th components of a vector representing a given document d is a numerical weight associated to the i -th feature w of the dictionary that occurs in d . Similarly, profiles are derived from the grouping of positive instances d in class C_i , i.e. $d \in C_i$.

Example-based are other types of classifiers, in which the incoming document d is used as a query against the training data (i.e. the set of training documents). Similarity between

d and class is evaluated as cumulative estimation between the input document and a portion of the training documents belonging to that class. The categories under which the training documents with the highest similarity are categorized, are considered as promising classification candidates for d . This approach is also referred as *document-centered* categorization. For both the above models a document is considered valid for a given class *iff* the similarity estimation overcomes established thresholds. The latter are parameters that adjust the trade-off between precision and recall.

2.1 The Problem of Feature Selection

Feature Selection techniques have been early introduced in order to limit the dimensionality of the feature space of text categorization problems. The native feature space consists of the unique terms (words or phrases) that occur in documents, which can be hundreds of thousands of terms even for a small text collection. This size prevents the applicability of many learning algorithms. Few neural models, for example, can handle such a large number of features usually mapped into input nodes.

Automatic feature selection methods foresee the removal of noninformative terms according to corpus statistics, and the construction of new (i.e. reduced or remapped) feature set. Common statistics parameters are: the *information gain* (e.g. [Yang and Pedersen, 1997]) aggressively reduces the document vocabulary, according to a naive Bayes model; a decision tree approach to select the most promising features wrt to a binary classification task; mutual information and a χ^2 statistic have been used to select features for input to neural networks; document clustering techniques estimating probabilistic "term strength"; inductive learning algorithms that derive features in disjunctive normal form.

As pointed out in [Yang and Pedersen, 1997] document frequency (DF), χ^2 and information gain provide the best selectors able to reduce the feature set cardinality and produce an increment of text classifier performances. The following equations describes four selectors among those experimented in [Yang and Pedersen, 1997]. They are based on both mutual information and χ^2 statistics:

$$I_{max}(f) = \max_i \{I(f, C_i)\}, I_{avg}(f) = \sum_i P_r(C_i) \cdot I(f, C_i)$$

$$\chi^2_{max}(f) = \max_i \{\chi^2(f, C_i)\}, \chi^2_{avg}(f) = \sum_i P_r(C_i) \cdot \chi^2(f, C_i)$$

where

- $P_r(C_i)$ is the probability of a generic document belonging to a class C_i , as observed in the training corpus
- f is a generic feature
- $I(f, C_i)$ is the mutual information between f and C_i ,
- $\chi^2(f, C_i)$ is the χ^2 value between f and C_i

After the ranking is derived, selection is carried out by removing the features characterized by the lowest scores (thresholding). Each of the above models produces a ranking of the different features f that is the same for all the classes: each of the above formulas suggests only one weight depending on all the classes. For example, the selector of a feature by

I_{avg} applies the average function to the set of $I(f, C_i)$ scores: every dependence on the i -th class disappear resulting in one single ranking. The same is true for χ_{max}^2 and χ_{avg}^2 .

Notice that this ranking, uniform throughout categories, may select features which are non globally informative but are enough relevant only for a given (or few) class(es) (e.g. the *max* or *avg*). The selection cannot take into account differences in the relevance among classes. Classes that are more generic, e.g. whose values of $I(f, C_i)$ (or χ^2) tend to be low, may result in a very poor profile, i.e. fewer number of selected features. This is in line with the observation in [Joachims, 1998] where the removal of features is suggested as a loss of important information, as the number of truly irrelevant features is negligible. Moreover, functions like *avg* are even more penalizing as they flatten the relevance of a single feature for each class to an "ideal" average value. Notice that this weakness is also reflected by the poorer results reported in [Yang and Pedersen, 1997].

In order to account for differences in the distribution of relevance throughout classes, we should depart from the idea of a uniform ranking. Features should be selected with respect to a single category. This can lead to retain features only when they are truly informative for some classes. Moreover a suitable class-based ranking is obtained, so that the feature scores (e.g. the mutual information $I(f, C_i)$) can be straightforwardly assumed as weights for the features in class C_i .

In next section an extension of the Rocchio formula aiming to obtain such desirable feature weights is presented.

2.2 Rocchio classifiers

The Rocchio classifier is a profile based classifier, presented in [Ittner *et al.*, 1995], which uses the Rocchio's formula for building class profiles. Given the set of training documents R_i classified under the topics C_i , the set \bar{R}_i of the documents not belonging to C_i , and given a document h and a feature f , the weight Ω_f^h of f in the profile of C_i is:

$$\Omega_f^i = \max \left\{ 0, \frac{\beta}{|R_i|} \sum_{d_h \in R_i} \omega_f^h - \frac{\gamma}{|\bar{R}_i|} \sum_{d_h \in \bar{R}_i} \omega_f^h \right\} \quad (1)$$

where ω_f^h represent the weights of features in documents¹. In Eq. 1 the parameters β and γ control the relative impact of positive and negative examples and determine the weight of f in the i -th profile. In [Ittner *et al.*, 1995], (1) has been firstly used with values $\beta = 16$ and $\gamma = 4$: the task was categorisation of low quality images. The success of these values possibly led to a wrong reuse of them in other fields [Cohen and Singer, 1996].

These parameters indeed greatly depend on the training corpus and different settings of their values produce a significant variation in performances. Poor performances have been obtained indeed in [Yang, 1999], and a wrong γ and β setting (maybe the original Ittner one) is a possible explanation. In [Joachims, 1998], the trial with a small set of values for β ($\{0, 0.1, 0.25, 0.5, 1.0\}$) is carried out and increased performance wrt those previously assessed by other authors are

¹Several methods are used to assign weights of a feature, as widely discussed in [Salton and Buckley, 1988]

obtained. However, the corresponding γ values are not mentioned. The impact of the adjustment of γ and β is significant if optimal values are systematically estimated from the training corpus. Experimental evidence will be further shown in Section 4.1.

Tuning Rocchio's formula parameters

As previously discussed, the Rocchio classifier strongly relies on the γ and β setting. However, the relevance of a feature deeply depends on the corpus characteristic and, in particular, on the differences among the training material for the different classes, i.e. size, the structure of topics, the style of documents, This varies very much across text collections and across the different classes within the same collection.

Notice that, in Equation 1, features with negative difference between positive and negative relevance are set to 0. This implies a discontinuous behavior of the Ω_f^i values around the 0. This aspect is crucial since the 0-valued features are irrelevant in the similarity estimation (i.e. they give a null contribution to the scalar product). This form of selection is rather smooth and allows to retain features that are selective only for some of the target classes. As a result, features are optimally used as they influence the similarity estimation for all and only the classes for which they are selective. In this way, the minimal set of truly irrelevant features (giving 0 values for all the classes) can be better captured and removed, in line with [Joachims, 1998].

Moreover, the γ and β setting that is fitted with respect to the classification performance has two main objectives:

- First, noise is drastically reduced by the Rocchio formula smoothing and without direct feature deletion.
- Second, the resulting ranking provides Rocchio-based scores that can be directly used as weights in the associated feature space. The higher is the positive evidence (wrt to the negative one) the higher is the relevance and this may vary for each target class.

Notice now that each category has its own set of relevant and irrelevant features and Eq. 1 depends for each class i on γ and β . Now if we assume the optimal values of these two parameters can be obtained by estimating their impact on the classification performance, nothing prevents us from driving this estimation independently for each class i . This will result in a vector of (γ_i, β_i) couples each one optimizing the performance of the classifier over the i -th class. Hereafter we will refer to this model as the *Rocchio* _{γ_i} classifier.

Notice that the proposed approach could converge to the traditional Rocchio weighting *if and only if* a single optimal value for γ and β is obtained, i.e. $\forall i \gamma_i = \gamma$ and $\beta_i = \beta$. This has not been the case as in Section 4.2 will be shown.

Finally, it has to be noticed that combined estimation of the two parameters is not required. For each class, we fixed one parameter (β_i indeed) and let γ_i vary until the optimal performance is reached. The weighting, ranking and selection scheme used in the for *Rocchio* _{γ_i} classifier is thus the following:

$$\Omega_f^i = \max \left\{ 0, \frac{1}{|R_i|} \sum_{d_h \in R_i} \omega_f^h - \frac{\gamma_i}{|\bar{R}_i|} \sum_{d_h \in \bar{R}_i} \omega_f^h \right\} \quad (2)$$

In our experiments, β has been set to 1, Equation 2 has been applied given the parameters γ_i that for each class C_i lead to the maximum breakeven point² of C_i .

3 The role of NLP in feature extraction

One of the aim of this work was to emphasize the role of linguistic information in the description (i.e. feature extraction) of different classes in a TC task. It is to be noticed that these latter are often characterized by sets of *typical* concepts usually expressed by multi-words expressions, i.e. linguistic structures synthesizing widely accepted definitions (e.g. "bond issues" in topics like "Finance or Stock Exchange"). These sets provide useful information to capture semantic aspects of a *topics*. The multi-word expressions are at least in two general classes useful for TC:

- Proper Nouns (PN), which usually do not bring much selective information in TC. Most named entities are locations, persons or artifacts and are rarely related to the semantics of a class. An evidence of this is discussed in [Basili *et al.*, 2000b] where PN removal is shown to improve performances.
- Terminological expressions, i.e. lemmatized phrase structures or single terms. Their detection results in a more precise set of features to be included in the target vector space.

The identification of linguistically motivated terminological structures usually requires external resources (thesauri or glossaries): as extensive repositories are costly to be developed and simply missing in most domains, an enumerative approach cannot be fully applied. Automatic methods for the derivation of terminological information from texts can thus play a key role in content sensitive text classification.

As terms embody domain specific knowledge we expect that their derivation from a specialized corpus can support the matching of features useful for text classification. Once terms specific to a given topics C_i are available (and they can be estimated from the training material for C_i), their matching in future texts d should strongly suggest classification of d in C_i .

Several methods for corpus-driven terminology extraction have been proposed (e.g. [Daille, 1994; Arppe, 1995; Basili *et al.*, 1997]). In this work, the terminology extractor described in [Basili *et al.*, 1997] has been adopted in the training phase. Each class (considered as a separate corpus) gives rise to a set of terms, T_i . When available, lemmatized phrase structures or single lemmas in T_i can be matched in future test documents. They are thus included in the final set of features of the target classifier.

Other features provided by linguistic processing capabilities are lemmas and their associated POS information able to capture word syntactic roles (e.g. *adjective*, *verb*, *noun*)³.

²It is the threshold values for which precision and recall coincide (see [Yang, 1999] for more details).

³These features have been built using the linguistic engine of the Trevi system, [Basili *et al.*, 2000a], where the interested reader can also find more details on the linguistic processing.

A further novel aspect of the classifier proposed in this paper is the application of the Equation 2 as a weighting scheme of the linguistically derived features. This allows to better separate the relevant information from the irrelevant one (possibly introduced by errors in the linguistic processing, e.g. wrong POS assignment). Finally, those irrelevant features, that are not necessarily produced via complex linguistic processing (e.g. single words), are correctly smoothed by Eq. 2 and this also helps in a more precise measurement of the NLP contribution.

4 Experimenting NLP-driven classification

The experiments have been carried out in two phases. First, we experimented Rocchio classifiers over a standard feature set, i.e. simple words. This serves two purposes. First, it provides the evaluation of the Breakeven Point reachable by a Rocchio classifier (via estimation of the suitable, but global, γ parameter). This allows a direct and consistent comparisons with the Rocchio models proposed in literature. Furthermore the first experiment also suggests the parameter setting that provides the best breakeven point of the extended *Rocchio* _{γ_i} model. In a second phase this optimal *Rocchio* _{γ_i} model has been experimented by feeding it with linguistic features. Comparative analysis of the two outcomes provides evidence of the role of this augmented information.

As a reference collection the Reuters, version 3, corpus prepared by Apté [Yang, 1999; Apté *et al.*, 1994] has been used. It will be hereafter referred as Reuters3. It includes 11,099 documents for 93 classes, with a fixed splitting between test and learning data (3,309 vs. 7,789). Every experiment thus allows direct comparisons with others models described in Section 5.

4.1 Deriving a baseline classifier

In these experiments, only words are taken as features and no other NLP facility has been applied in agreement with other methods described in literature. The feature weight in a document is the usual product between the logarithm of the feature frequency (inside the document) and the associated inverse document frequency. The best Rocchio classifier performances has been derived by systematically setting different values of γ and optimizing performance (i.e. the breakeven point, BEP). A sequence of classifiers has thus been obtained. In Figure 1 the plot of BEPs with respect to γ is shown. The two plots refer to two different feature sets: *SimpleFeatures* refers to single words while *LingFeatures* refers to the model using all the available linguistic information (see Sect. 4.2). First of all, performances depend strongly on the parameters. The best performance of the *SimpleFeatures* model is reached with $\gamma = 6$ (and $\beta=1$): these values significantly differ from the $\gamma=16$ and $\beta=4$ used elsewhere. Second, significant higher performances characterize the language-driven model for all the γ values: this shows an inherent superiority of the source linguistic information. However, when a single γ for all the classes is used, a suitable adjustment let the *SimpleFeatures* model to approximate the behaviour of the linguistic one. This is no longer true when more selective estimation of the parameter (i.e. $\gamma_i \forall i$) is applied.

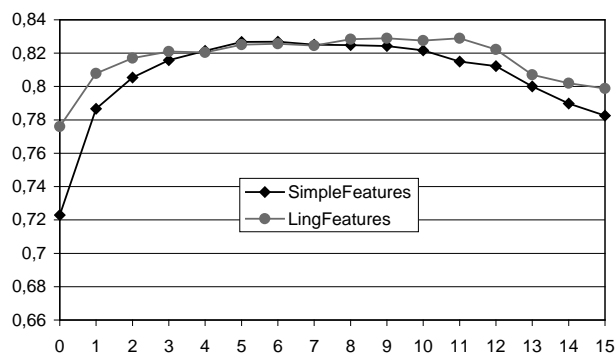


Figure 1: Break-even point performances of the Rocchio classifier according to different γ values

In a second experiment indeed the parameter estimation process has been individually applied to each class i , and the optimal sequence of γ_i values has been obtained. In the Table 1 are shown the performances of three Rocchio classifiers: simple Rocchio (as in [Ittner *et al.*, 1995], with $\gamma = 4/16$), as well as *Rocchio* $\gamma = 6$ and *Rocchio* $_{\gamma_i}$ characterized by one global parameter and individual parameters, respectively. The three tests have been carried out by using only simple

Table 1: Breakeven points of three Rocchio-based models on Reuters3

<i>Rocchio</i> $_{\gamma_i}$	<i>Rocchio</i> $\gamma = 6$	<i>Rocchio</i>
83.82%	82.61%	75% - 79.9%

words as features, in line with traditional techniques. Notice that both the optimized *Rocchio* $\gamma = 6$ and *Rocchio* $_{\gamma_i}$ models, proposed in this paper, outperform all the best results obtained in literature for Rocchio classifiers (e.g. [Joachims, 1998; Cohen and Singer, 1996]).

4.2 Comparing different NLP-based classifiers

Once the best weighting technique has been assessed as the optimal estimation of parameters γ_i in the previous experiments, it is possible to selectively measure the contribution given by NLP. In fact an independent baseline model, with minimal noisy information, (i.e. the *Rocchio* $_{\gamma_i}$ model in Table 1), is used contrastively to correctly measure the contribution brought by the augmented features. In the following experiment, the novel sets of features described in Section 3 have been added to the standard set. They consist of:

- Proper Nouns (+PN or -PN if recognition in text is followed by removal during TC training)
- Terminological Expressions (+TE)
- Lemmas (-POS)
- Lemmas augmented with their POS tags in context (+POS)

Terminological expressions have been firstly derived from the training material of one class: for example, in the class *acq* (i.e. *Mergers and Acquisition*) of the Reuters3, among the

9,650 different features about 1,688 are represented by terminological expressions or complex Proper Nouns (17%). The *Rocchio* $_{\gamma_i}$ model has been selectively applied to three linguistic features set: the first includes only the lemmas associated to the POS tag (+POS), the second lemmas only (-POS), and the third Proper Nouns and Terminological Expressions (+POS+PN+TE).

In Table 2 is reported the BEP of the three feature sets: the comparison is against the baseline, i.e. the best non linguistic result of Tab. 1, although reestimation of the parameters has been carried out (as shown by Fig. 1).

We observe that both POS tag (column 4 vs column 3) and terminological expressions (column 3 vs column 1) produce improvements when included as features. Moreover PNs seems not to bring more information than POS tags, as column 2 suggests. The best model is the one using all the linguistic features provided by NLP. This increases performance (> 1%) which is not negligible if considering the very high baseline.

Table 2: Breakeven points of *Rocchio* $_{\gamma_i}$ on three feature set provides with NLP applied to Reuters version 3

Base-Line	+POS-PN	+PN+TE	+PN+TE+POS
83.82%	83.86%	84.48%	84.94%

5 Discussion

In Table 3 the performances of the most successful methods proposed in literature are reported. Some of their distinctive aspects are here briefly summarized. Support Vector Machines *SVM* recently proposed in [Joachims, 1999] is based on the structural risk minimisation principle. It uses quadratic programming technique for finding a surface that "best" separates the data points (the representation of training documents in the vector space model) in two classes. *K*-Nearest Neighbour is an example-based classifier, [Yang and Liu, 1999], making use of document to document similarity estimation that selects a class for a document through a *k*-nearest heuristics. RIPPER [Cohen and Singer, 1996] uses an extended notion of profile, by learning contexts that are positively correlated with the target classes. A machine learning algorithms allows the "contexts" of a word w to affect how (or whether) presence/absence of w contribute actually to a classification. CLASSI is a system that uses a neural network-based approach to text categorization [H.T. Ng, 1997]. The basic units of the network are only perceptrons. Dtree [Quinlan, 1986] is a system based decision trees. The Dtree model allows to select relevant words (i.e. features), according to an information gain criterion. CHARADE [I. Moulinier and Ganascia, 1996] and SWAP1 [Apté *et al.*, 1994] use machine learning algorithms to inductively extract Disjunctive Normal Form rules from training documents. Sleeping Experts (EXPERTS) [Cohen and Singer, 1996] are learning algorithms that works on-line. They reduce the computation complexity of the training phase for large applications updating incrementally the weights of n -gram phrases.

Two major conclusions can be drawn. First of all the parameter estimation proposed in this paper is a significant improvement with respect to other proposed uses of the Rocchio formula. The application of this method over crude fea-

Table 3: BEP of best classifiers on Reuters3 - Revised

<i>SVM</i>	KNN	<i>Rocchio</i> _{γ_i} +NLP	<i>Rocchio</i> _{γ_i}
85.99%	85.67%	84.94%	83.82%
RIPPER	CLASSI	DTREE	SWAPI
80%	80%	79%	79%
CHARADE	EXPERT	Rocchio	Naive Bayes
78%	76%	82.61% (75% - 79.9%)	71%-79%

ture sets (i.e. simple words and without any selection) improve significantly with respect to the best obtained Rocchio methods (83.82% vs 79.9%). This weighting scheme is a robust filtering technique for sparse data in the training corpus. It has been suitably applied to derive the baseline figures for contrastive analysis of the role of linguistic features.

The comparative evaluation of simpler feature sets with linguistically motivated information (i.e. POS tagged lemmas and terminological information) suggests the superiority of the latter. This is mainly due the adoption of the optimal selection and weighting method proposed. It provides a systematic way to employ the source linguistic information. It has to be noticed that in the set of 9,650 features (including linguistic ones) derived from the training material of the Reuters3 *acq* category, only 4,957 ($\sim 51.3\%$) assumes a weight greater than 0 after γ_{acq} optimization is carried out. Notice that the use of a single (global) γ value over linguistic features (i.e. +POS+PN+TE), shown in Fig. 1 (*LingFeature* plot), reaches a best BEP of about 0.828: this is improved of more than 2% in BEP when selective γ_i setting is applied (Tab. 2). This form of weighting is thus responsible for an optimal employment of linguistic information that is, by its nature, often affected by data sparseness and noise.

6 Conclusion

In this paper a new model able to exactly measure the contribution given by NLP has been designed and experimented. It brings significant evidence of the role of natural language processing techniques in the specific TC area of IR. The benefits of NLP methods are the efficient extraction of linguistically motivated complex features (including multi-word patterns). A novel weighting method has been also proposed. It provides a systematic feature selection functionality with a systematic estimation of the γ_i parameters in the Rocchio formula. The method is robust and effective wrt noise as analysis over non linguistic feature sets demonstrates. This gave us the possibility of focusing on the measurement of relevance of NLP derived features. The *Rocchio* _{γ_i} applied to linguistic material supports thus a computationally efficient classification (typical of purely statistical models) and produces performances (about 85%) comparable with the best (but computationally more expensive) classifiers (e.g. KNN and SVM).

References

[Apté et al., 1994] Chidanand Apté, Fred Damerau, and Sholom Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[Arppe, 1995] A. Arppe. Term extraction from unrestricted text. In *NODALIDA*, 1995.

[Basili et al., 1997] R. Basili, G. De Rossi, Pazienza, and M.T. Inducing terminology for lexical acquisition. In *Proceedings of the Second Conference on Empirical Methods in NLP*, Providence, USA, 1997.

[Basili et al., 2000a] R. Basili, L. Mazzucchelli, and M.T. Pazienza. An adaptive and distributed framework for advanced ir. In *In proceeding of 6th RIAO Conference, Content-Based Multimedia Information Access, Collge de France, Paris, France*, 2000.

[Basili et al., 2000b] R. Basili, A. Moschitti, and M.T. Pazienza. Language sensitive text classification. In *In proceeding of 6th RIAO Conference, Content-Based Multimedia Information Access, Collge de France, Paris, France*, 2000.

[Cohen and Singer, 1996] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 96)*, pages 12–20, 1996.

[Daille, 1994] B. Daille. Study and implementation of combined techniques for automatic extraction of terminology. In *The Balancing Act Workshop of the 32nd Annual Meeting of the ACL*, 1994.

[Grefenstette, 1997] Gregory Grefenstette. Short queries linguistic expansion techniques: Palliating one-word queries by providing intermediate structures to text. In M.T. Pazienza, editor, *Information Extraction*. Springer Verlag, Berlin, 1997.

[H.T. Ng, 1997] K.L. Low H.T. Ng, W.B. Goh. Features selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th ACM SIGIR Conference*, pages 67–73, 1997.

[I. Moulinier and Ganascia, 1996] G. Raskinis I. Moulinier and J. Ganascia. Text categorization: a symbolic approach. In *In Proceedings of the 5th Annual Symposium on Document Analysis and Information Retrieval*, 1996.

[Ittner et al., 1995] David J. Ittner, David D. Lewis, and David D. Ahn. Text categorization of low quality images. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, US, 1995.

[Joachims, 1999] T. Joachims. Transductive inference for text classification using support vector machines. In I. Bratko and S. Dzeroski editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, 1999.

[Joachims, 1998] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *In Proceedings of ECML-98*, pages 137–142, 1998.

[Lewis et al., 1996] David D. Lewis, Robert E. Schapiro, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of 19th ACM SIGIR-96*, pages 298–306, Zürich, CH, 1996.

[Quinlan, 1986] J.R. Quinlan. Induction of decision trees. In *Machine Learning*, pages 81–106, 1986.

[Salton and Buckley, 1988] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[Salton, 1991] G. Salton. Development in automatic text retrieval. *Science*, 253:974–980, 1991.

[Yang and Liu, 1999] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *In Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[Yang and Pedersen, 1997] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of 14th ICML-97*, pages 412–420, Nashville, US, 1997.

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

NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

NATURAL LANGUAGE EXPLANATION
AND ARGUMENTATION

Dialog-driven Adaptation of Explanations of Proofs

Armin Fiedler

Universität des Saarlandes, FR Informatik
Postfach 15 11 50, D-66041 Saarbrücken, Germany
afiedler@cs.uni-sb.de

Abstract

In order to generate high quality explanations in mathematical domains, the presentation must be adapted to the knowledge of the intended audience. Most proof presentation systems only communicate proofs on a fixed degree of abstraction independently of the addressee's knowledge. In this paper, we shall present the proof explanation system *Prex*. Based on assumptions about the addressee's knowledge, its dialog planner chooses a degree of abstraction for each proof step to be explained. In reaction to the user's interactions, which are allowed at any time, it enters clarification dialogs to revise its user model and to adapt the explanation.

1 Introduction

A person who explains to another person a logical line of reasoning adapts his explanations to the addressee's knowledge. A computer program designed to take over the explaining part should also adopt this principle.

Assorted systems take into account the intended audience's knowledge in the generation of explanations (see e.g., [Cawsey, 1990; Paris, 1991; Wahlster *et al.*, 1993; Horacek, 1997]). Most of them adapt to the addressee by choosing between different discourse strategies. Feedback from the user himself can be an important source for user modeling. [Moore and Swartout, 1991] presents a context-sensitive explanation facility for expert systems that, on the one hand, allows the user to ask follow-up questions and, on the other hand, actively seeks feedback from the user to determine whether the explanations are satisfactory. [Mooney *et al.*, 1991] emphasizes that the user must be able to interrupt the explanation system at any time.

Whereas a mathematician communicates a proof on a level of abstraction that is tailored to the audience and reacts to the audience's needs, most proof presentation systems such as *PROVERB* [Huang and Fiedler, 1997] verbalize proofs on a fixed degree of abstraction given by the initial representation of the proof without allowing for user interaction.

Drawing on results from cognitive science, we have been developing an *interactive proof explanation system*, called *Prex* (for *proof explainer*), which adapts its explanations to the user's assumed expertise. The system explains each proof

step of a proof on the most abstract level that the user is assumed to know. While the explanation is in progress, the user can interrupt *Prex* anytime, when he is not satisfied by the current explanation. Analyzing the user's interaction and entering a clarification dialog when needed, the system tries to identify the reason why the explanation was not satisfactory and re-plans a better adapted explanation, for example, by switching to another level of abstraction.

Hence, driven by the dialog, *Prex* adapts its explanations to the user in reaction to his interactions. However, in the current experimental stage, only a small set of interactions is allowed.

In this paper, we shall first introduce proofs as the domain objects that are subject to explanation in Section 2. Next, in Section 3, we shall give an overview of the architecture of *Prex*. Then, in Section 4, we shall describe the dialog planner in more detail. Section 5 is devoted to example dialogs.

2 The Domain

The objects of our domain that are to be explained are proofs of mathematical theorems. An example for a proof is given below. Each line consists of four elements (label, antecedent, succedent, and justification) and describes a node of the proof. The *label* is used as a reference for the node. The *antecedent* is a list of labels denoting the hypotheses under which the formula in the node, the *succedent*, holds. This relation between antecedent and succedent is denoted by \vdash .

Label	Antecedent	Succedent	Justification
L_0		$\vdash a \in U \vee a \in V$	J_0
H_1	H_1	$\vdash a \in U$	HYP
L_1	H_1	$\vdash a \in U \cup V$	Def \cup (H_1)
H_2	H_2	$\vdash a \in V$	HYP
L_2	H_2	$\vdash a \in U \cup V$	Def \cup (H_2)
L_3		$\vdash a \in U \cup V$	\cup -Lemma(L_0) CASE(L_0, L_1, L_2)

We call $\Delta \vdash \varphi$ the *fact* in the node. The proof of the fact in the node is given by its *justification*. A justification consists of a rule and a list of labels, the *premises* of the node. J_i denotes an unspecified justification. HYP and Def \cup stand for a hypothesis and the definition of \cup , respectively. L_3 has two justifications on different levels of abstraction: the less abstract justification with the inference rule CASE (i.e., the rule for case analyses) and the more abstract justification with the

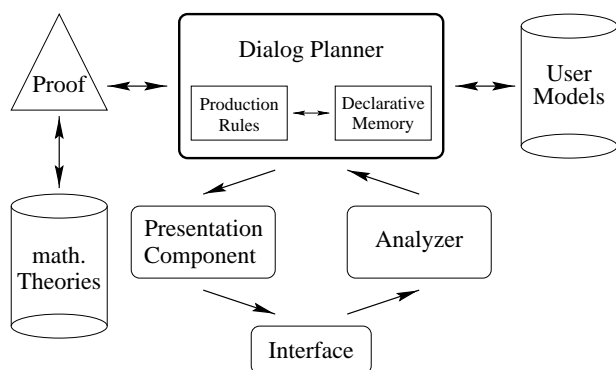


Figure 1: The Architecture of *P.rex*

inference rule \cup -Lemma, which stands for an already proven lemma about a property of \cup . By agreement, if a node has more than one justification, these are sorted from most abstract to least abstract.

The proof is as follows: From $a \in U \vee a \in V$ we can conclude that $a \in U \cup V$ by the \cup -Lemma. If we do not know the \cup -Lemma, we can come to the conclusion by considering the case analysis with the cases that $a \in U$ or $a \in V$, respectively. In each case, we can derive that $a \in U \cup V$ by the definition of \cup .

3 The Architecture of *P.rex*

P.rex is a generic explanation system that can be connected to different theorem provers. An overview of its architecture is provided in Figure 1.

A formal language for specifying proofs and mathematical theories is the interface by which theorem provers can be connected to *P.rex*. **Mathematical theories** are organized in a hierarchical knowledge base. Each theory in it may contain axioms, definitions, theorems along with proofs, as well as proof methods, and control rules how to apply proof methods.

A **proof** of a theorem can be represented hierarchically such that it makes explicit the various levels of abstraction by providing several justifications for a single proof node, where each justification belongs to a different level of abstraction.

The central component of the system is the **dialog planner**. It is implemented in ACT-R [Anderson and Lebiere, 1998], a goal-directed production system that aims to model human cognition. ACT-R has two types of knowledge bases, or *memories*, to store permanent knowledge in: declarative and procedural representations of knowledge are explicitly separated into the declarative memory and the procedural production rule base, but are intimately connected.

Procedural knowledge is represented in production rules (or simply: *productions*) whose conditions and actions are defined in terms of declarative structures. A production can only apply if its conditions are satisfied by the knowledge currently available in the declarative memory. An item in the declarative memory is annotated with an activation that influences its retrieval. The application of a production modifies the declarative memory, or it results in an observable event.

To allow for a goal-oriented behavior of the system, ACT-R manages goals in a goal stack. The current goal is that on the top of the stack. Only productions that match the current goal are applicable.

The plan operators of the dialog planner are defined in terms of productions and the discourse structure is represented in the declarative memory. Since presumed declarative and procedural knowledge of the user is encoded in the declarative memory and the production rule base, respectively, the dialog planner is modeling the user.

In order to explain a particular proof, the dialog planner first assumes the user's supposed cognitive state by updating its declarative and procedural memories, which were recorded during a previous session. An individual model for each user persists between the sessions. The individual user models are stored in the database of **user models**. Each user model contains assumptions on the knowledge of the user that are relevant to proof explanation. In particular, it makes assumptions on which mathematical theories the user knows, which definitions, proofs, proof methods and mathematical facts he knows, and which productions he has already learned.

After updating the declarative and procedural memories, the dialog planner sets the global goal to show the conclusion of the proof's theorem. ACT-R tries to fulfill this goal by successively applying productions that decompose or fulfill goals. Thereby, the dialog planner not only produces a dialog plan (cf. Section 4.1), but also traces the user's cognitive states in the course of the explanation. This allows the system both to always choose an explanation adapted to the user (cf. Section 4.3), and to react to the user's interactions in a flexible way: The dialog planner analyzes the interaction in terms of applications of productions. Then it plans an appropriate response (cf. Section 4.4).

The dialog plan produced by the dialog planner is passed on to the **presentation component**. Currently, we use a derivative of *PROVERB*'s micro-planner [Huang and Fiedler, 1997] to plan the scope and internal structure of the sentences, which are then realized by the syntactic generator TAG-GEN [Kilger and Finkler, 1995].

The uttered sentences are finally displayed on the **interface**, which also allows the user to enter remarks, requests and questions. An **analyzer** receives the user's interactions and passes them on to the dialog planner. In the current experimental stage, we use a simplistic analyzer that understands a small set of predefined interactions.

4 Discourse Planning

In the community of NLG, there is a broad consensus that it is appropriate to generate natural language in three major steps [Reiter, 1994; Cahill *et al.*, 1999]. First, a *macro-planner* (*text planner*) determines what to say, that is, content and order of the information to be conveyed. Then, a *micro-planner* (*sentence planner*) determines how to say it, that is, it plans the scope and the internal structure of the sentences. Finally, a *realizer* (*surface generator*) produces the surface text. In this classification, the dialog planner is a macro-planner for managing dialogs.

The dialog planner of *P.rex* plans the dialog by building a representation of the structure of the discourse that includes speech acts as well as relations among them. The discourse structure is represented in the declarative memory. The plan operators are defined as productions.

4.1 Discourse Structure

Drawing on *Rhetorical Structure Theory (RST)* [Mann and Thompson, 1987], Hovy argues in favor of a single tree to represent a discourse [Hovy, 1993]. He considers a discourse as a structured collection of clauses, which are grouped into segments by their semantic relatedness. The discourse structure is expressed by the nesting of segments within each other according to specific relationships (i.e., RST relations). Similarly to Hovy's approach, we describe a discourse by a *discourse structure tree*, where each node corresponds to a segment of the discourse. The speech acts, which correspond to minimal discourse segments, are represented in the leaves. We achieve a linearization of the speech acts by traversing the discourse structure tree depth-first from left to right.

4.2 Speech Acts

Speech acts are the primitive actions planned by the dialog planner. They represent frozen rhetorical relations between exchangeable semantic entities. The semantic entities are represented as arguments to the rhetorical relation in the speech act. Each speech act can always be realized by a single sentence. We use speech acts in *P.rex* not only to represent utterances that are produced by the system, but also to represent utterances from the user in the discourse.

We distinguish two major classes of speech acts. First, *mathematical communicative acts (MCAs)* are employed to convey mathematical concepts or derivations. MCAs suffice for those parts of the discourse, where the initiative is taken by the system. Second, *interpersonal communicative acts (ICAs)* serve the dialog, where both the system and the user alternately take over the active role.

Mathematical Communicative Acts

Mathematical communicative acts (MCAs) are speech acts that convey mathematical concepts or derivations. Our class of MCAs was originally derived from *PROVERB*'s PCAs [Huang, 1994], but has been substantially reorganized and extended. We distinguish two classes of MCAs:

- *Derivational MCAs* convey steps of the derivation, which are logically necessary. Failing to produce a derivational MCA makes the presentation logically incorrect. The following is an example for a derivational MCA given with a possible verbalization:

```
(Derive :Reasons (a ∈ F ∨ a ∈ G)
          :Conclusion a ∈ F ∪ G
          :Method U-Lemma)
```

“Since $a \in F$ or $a \in G$, $a \in F \cup G$ by the U-Lemma.”

- *Explanatory MCAs* comment on the steps of a derivation or give information about the structure of a derivation. This information is logically unnecessary, that is, omission leaves the derivation logically correct. However, inclusion of explanatory MCAs makes it much easier for

the addressee to understand the derivations, since these comments keep him oriented. For example, an MCA of type Case introduces a case in a case analysis:

(Case :Number 1 :Hypothesis $a \in F$)

“Case 1: Let $a \in F$.”

Interpersonal Communicative Acts

MCAs, which only convey information to the dialog partner without prompting any interaction, suffice to present mathematical facts and derivations in a monolog. To allow for dialogs we also need *interpersonal communicative acts (ICAs)*, which are employed for mixed-initiative, interpersonal communication. In our taxonomization we distinguish four classes of ICAs: questions, requests, acknowledgments and notifications. Note that the user never enters speech acts directly into the system. Instead, the user's utterances are interpreted by the analyzer and mapped into the corresponding speech acts.

ICAs are especially important to allow for clarification dialogs. If the system failed to successfully communicate a derivation, it starts a clarification dialog to detect the reason for the failure. Then, it can re-plan the previously failed part of the presentation and double-check that the user understood the derivation. We shall come back to this issue in Section 4.4.

4.3 Plan Operators

Operational knowledge concerning the presentation is encoded as productions. Each production either fulfills the current goal directly or splits it into subgoals. Let us assume that the following nodes are in the current proof:

Label	Antecedent	Succedent	Justification
P_1	Δ_1	$\vdash \varphi_1$	J_1
		\vdots	
P_n	Δ_n	$\vdash \varphi_n$	J_n
C	Γ	$\vdash \psi$	$R(P_1, \dots, P_n)$

An example for a production is:

```
(P1) IF    the current goal is to show  $\Gamma \vdash \psi$ 
           and  $R$  is the most abstract known rule justifying
           the current goal
           and  $\Delta_1 \vdash \varphi_1, \dots, \Delta_n \vdash \varphi_n$  are known
THEN      produce MCA
           (Derive :Reasons ( $\varphi_1, \dots, \varphi_n$ )
             :Conclusion  $\psi$  :Method  $R$ )
           insert it in the discourse structure tree
           and pop the current goal
```

By producing the MCA the current goal is fulfilled and can be popped from the goal stack. An example for a production decomposing the current goal into several subgoals is:

```
(P2) IF    the current goal is to show  $\Gamma \vdash \psi$ 
           and  $R$  is the most abstract known rule justifying
           the current goal
           and  $\Phi = \{\varphi_i \mid \Delta_i \vdash \varphi_i \text{ is unknown for } 1 \leq i \leq n\} \neq \emptyset$ 
THEN      for each  $\varphi_i \in \Phi$ , push the goal to show  $\Delta_i \vdash \varphi_i$ 
```

Note that the conditions of (P1) and (P2) only differ in the knowledge of the premises φ_i for rule R . (P2) introduces the subgoals to prove the unknown premises in Φ . As soon as those are derived, (P1) can apply and derive the conclusion.

Hence, (P1) and (P2) in principle suffice to plan the presentation of a proof starting from the conclusion and traversing the proof tree towards its hypotheses. However, certain proof situations call for a special treatment. Assume that the following nodes are in the current proof:

Label	Antecedent	Succedent	Justification
P_0	Γ	$\vdash \varphi_1 \vee \varphi_2$	J_0
H_1	H_1	$\vdash \varphi_1$	HYP
P_1	Γ, H_1	$\vdash \psi$	J_1
H_2	H_2	$\vdash \varphi_2$	HYP
P_2	Γ, H_2	$\vdash \psi$	J_2
C	Γ	$\vdash \psi$	CASE(P_0, P_1, P_2)

A specific production managing such a case analysis is the following:

(P3) IF the current goal is to show $\Gamma \vdash \psi$
and CASE is the most abstract known rule justifying the current goal
and $\Gamma \vdash \varphi_1 \vee \varphi_2$ is known
and $\Gamma, H_1 \vdash \psi$ and $\Gamma, H_2 \vdash \psi$ are unknown
THEN push the goals to show $\Gamma, H_1 \vdash \psi$ and $\Gamma, H_2 \vdash \psi$
and produce MCA
(Case-Analysis :Goal ψ
:Cases (φ_1, φ_2))
and insert it in the discourse structure tree

This production introduces new subgoals and motivates them by producing the MCA.

Since more specific productions such as (P3) treat common communicative standards used in mathematical presentations, they are preferred to more general ones. Note that the productions ensure that only those inference rules are selected for the explanation that are known to the user.

4.4 User Interaction

The ability for user interaction is an important feature of explanation systems. [Moore and Swartout, 1991] presents a context-sensitive explanation facility for expert systems that, on the one hand, allows the user to ask follow-up questions and, on the other hand, actively seeks feedback from the user to determine whether the explanations are satisfactory. [Mooney *et al.*, 1991] emphasizes that the user must be able to interrupt the explanation system at any time.

In *Prerex*, the user can interact with the system at any time. When the system is idle—for example, after starting it or after completion of an explanation—it waits for the user to tell it the next task. During an explanation, *Prerex* checks after each production cycle if the user wishes to interrupt the current explanation. Each interaction is analyzed by the analyzer and passed on to the dialog planner as a speech act, which is included in the current discourse structure tree.

We allow for three types of user interaction in *Prerex*: A *command* tells the system to fulfill a certain task, such as explaining a proof. An *interruption* interrupts the system to inform it that an explanation is not satisfactory or that the user wants to insert a different task. In clarification dialogs, finally, the user is prompted to give *answers* to questions that *Prerex* asks when it cannot identify a unique task to fulfill. In this paper, we shall concentrate on interruptions.

Interruptions

The user can interrupt *Prerex* anytime to enter a new command or to complain about the current explanation. The following speech acts are examples for messages that can be used to interrupt the system:

(too-detailed :Conclusion C)
The explanation of the step leading to C is too detailed, that is, the step should be explained at a more abstract level.
(too-difficult :Conclusion C)
The explanation of the step leading to C is too difficult, that is, the step should be explained in more detail.

The Reaction to too-detailed

When the user complains that the derivation of a conclusion C was too detailed, the dialog planner checks if there is a higher level of abstraction on which C can be shown. If so, the corresponding higher level inference rule is marked as known, so that it is available for future explanations. Then, the explanation of the derivation of C is re-planned. Otherwise, the dialog planner informs the user, that there is no higher level available for presentation. This reaction of the system is encoded in the following two productions:

(P4) IF the user message is
(too-detailed :Conclusion C)
and the inference rule R was used to justify C
and there is an inference rule R' justifying C that is more abstract than R
THEN mark R' as known by the user
and push the goal to show C
(P5) IF the user message is
(too-detailed :Conclusion C)
and the inference rule R was used to justify C
and there is no inference rule R' justifying C that is more abstract than R
THEN produce ICA
(Most-abstract-available :Rule R)
and insert it in the discourse structure tree

An example dialog where the user complained that the original explanation of a proof was too detailed shall be given in Example 1 in Section 5.

The Reaction to too-difficult

When the user complains that the derivation of a conclusion C was too difficult, the dialog planner enters a clarification dialog to find out which part of the explanation failed to remedy this failure. The control of the behavior of the dialog planner is displayed in Figure 2. Note that every arrow in the figure corresponds to a production such that we cannot give the productions here due to space restrictions.

To elucidate the diagram in Figure 2, an example dialog where the user complained that the original explanation of a proof was too difficult shall be given in Example 2 in the following section.

5 Example Dialogs

Let us examine more closely how *Prerex* plans the discourse with the help of two example dialogs. In both dialogs, *Prerex*

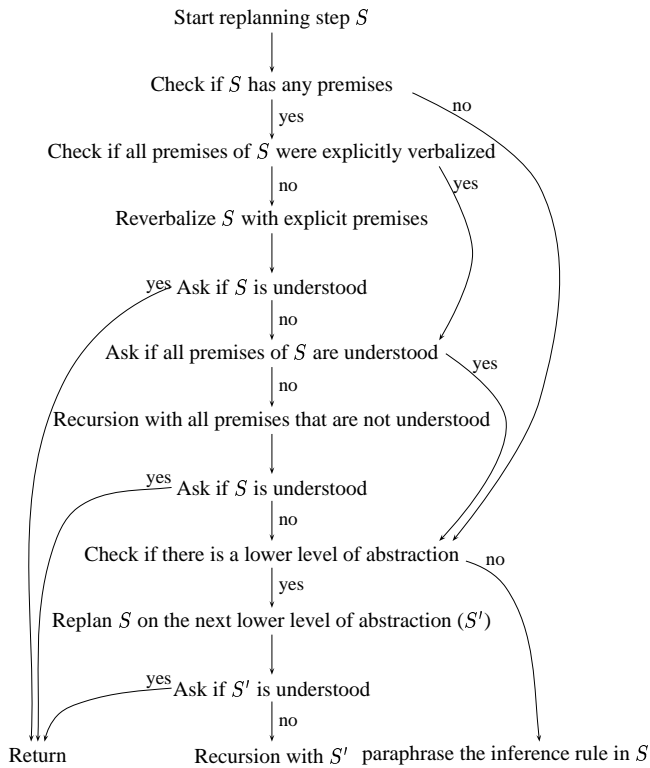


Figure 2: The reaction of the dialog planner if a step S was too difficult.

explains the proof given in Figure 3. Note that this proof consists of two similarly proved parts with L_3 and L_7 as roots, respectively.

Example 1 Let us consider the following situation:

- The current goal is to show the fact in L_3 . The next goal on the stack is to show the fact in L_7 .
- The rules HYP, CASE, and Def \cup are known, the rule \cup -Lemma is unknown.
- The facts in L_0 and L_4 are known, the facts in $H_1, L_1, H_2, L_2, H_5, L_5, H_6$, and L_6 are unknown.

Since CASE is the most abstract known rule justifying the current goal, both decomposing productions (P2) and (P3) are applicable. Recall that more specific productions are preferred to more general ones. Therefore, the dialog planner chooses (P3) for the explanation, thus producing the MCA

(Case-Analysis :Goal $a \in U \cup V$
:Cases ($a \in U, a \in V$))

which can be realized as “To prove $a \in U \cup V$ let us consider the cases that $a \in U$ and $a \in V$,” and then explains both cases.

Suppose now that the user interrupts the system throwing in that the presentation is too detailed. Then, the analyzer passes the speech act (too-detailed :Conclusion $a \in U \cup V$) to the dialog planner. Since the inference rule \cup -Lemma, which is more abstract than CASE, also justifies the conclusion, production (P4) applies. Hence, the inference rule \cup -Lemma is marked as known and the goal to show

Label	Antecedent	Succedent	Justification
L_0		$\vdash a \in U \vee a \in V$	J_0
H_1	H_1	$\vdash a \in U$	HYP
L_1	H_1	$\vdash a \in U \cup V$	Def \cup (H_1)
H_2	H_2	$\vdash a \in V$	HYP
L_2	H_2	$\vdash a \in U \cup V$	Def \cup (H_2)
L_3		$\vdash a \in U \cup V$	\cup -Lemma(L_0)
			CASE(L_0, L_1, L_2)
L_4		$\vdash a \in F \vee a \in G$	J_4
H_5	H_5	$\vdash a \in F$	HYP
L_5	H_5	$\vdash a \in F \cup G$	Def \cup (H_5)
H_6	H_6	$\vdash a \in G$	HYP
L_6	H_6	$\vdash a \in F \cup G$	Def \cup (H_6)
L_7		$\vdash a \in F \cup G$	\cup -Lemma(L_4)
			CASE(L_4, L_5, L_6)

Figure 3: A proof to be explained by *Prer*.

the fact in L_3 is again pushed onto the goal stack. Then, (P1) is the only applicable production. Since \cup -Lemma is more abstract than CASE and both are known, it is chosen to instantiate (P1). Hence, the dialog planner produces the MCA

(Derive :Reasons ($a \in U \vee a \in V$)
:Conclusion $a \in U \cup V$
:Method \cup -Lemma)

which can be verbalized as “Since $a \in U$ or $a \in V$, $a \in U \cup V$ by the \cup -Lemma.”

Since \cup -Lemma is now marked as known by the user, it can also be used for presentation in subsequent situations, for example, when L_7 is to be shown. The whole dialog takes place as follows:

Prer: In order to prove that $a \in U \cup V$ let us consider the following cases.

Case 1: Let $a \in U$. Then $a \in U \cup V$ by the definition of \cup .

Case 2: Let $a \in V$. That implies that $a \in U \cup V$ by the definition of \cup .

User: This derivation is too detailed.

Prer: Since $a \in U$ or $a \in V$, $a \in U \cup V$ by the \cup -Lemma. Since $a \in F$ or $a \in G$, $a \in F \cup G$ by the \cup -Lemma.

□

To elucidate the behavior of the dialog planner as depicted in Figure 2, let us examine the following example:

Example 2 We now consider the following situation:

- The current goal is to show the fact in L_3 . The next goal on the stack is to show the fact in L_7 .
- the rules HYP, CASE, Def \cup , and \cup -Lemma are known,
- The facts in L_0 and L_4 are known, the facts in $H_1, L_1, H_2, L_2, H_5, L_5, H_6$, and L_6 are unknown.

This time, the only applicable production is (P1). Since \cup -Lemma is more abstract than CASE and both are known, it is chosen to instantiate (P1). Hence, the dialog planner produces the MCA

(Derive :Reasons ($a \in U \vee a \in V$)
:Conclusion $a \in U \cup V$
:Method \cup -Lemma)

which can be verbalized as “Since $a \in U$ or $a \in V$, $a \in U \cup V$ by the \cup -Lemma.”

Suppose now that the user points to this utterance and interrupts *P.r*ex throwing in that this step was too difficult. The analyzer translates the user’s interaction to the speech act (`too-difficult :Conclusion $a \in U \cup V$`). Now, the dialog planner enters the clarification dialog as displayed in Figure 2. Since all premises were explicitly mentioned, the system does not reverb-alize the step, but asks whether all premises are understood, what the user affirms. Hence, the system checks whether there is a lower level of abstraction, at which the step can be presented. Since this is the case, *P.r*ex replans the explanation of the step by marking the inference rule \cup -Lemma as unknown and pushing the goal to show the fact in L_3 onto the goal stack. Now, (P1) is not applicable, since \cup -Lemma is unknown, but (P2) and (P3) are applicable. Hence, the system proceeds as in the beginning of Example 1 and verbalizes the case analysis. Since \cup -Lemma is unknown, it is not used in subsequent situations either. Therefore, L_7 is also explained by a case analysis. The whole dialog takes place as follows:

P.rex: Since $a \in U$ or $a \in V$, $a \in U \cup V$ by the \cup -Lemma.

User: This step is too difficult.

P.rex: Do you understand the premises?

User: Yes.

P.rex: In order to prove that $a \in U \cup V$ let us consider the following cases.

Case 1: Let $a \in U$. That leads to $a \in U \cup V$ by the definition of \cup .

Case 2: Let $a \in V$. Then $a \in U \cup V$ by the definition of \cup .

Do you understand this step?

User: Yes.

P.rex: In order to prove that $a \in F \cup G$ let us consider the following cases.

Case 1: Let $a \in F$. Therefore $a \in F \cup G$ by the definition of \cup .

Case 2: Let $a \in G$. Then $a \in F \cup G$ by the definition of \cup .

□

6 Conclusion

In this paper, we presented the dialog planner of the proof explanation system *P.r*ex. Based on assumptions about the addressee’s knowledge (e.g., which facts does he know, which definitions, lemmas, etc.), the dialog planner chooses a degree of abstraction for each proof step to be explained. In reaction to the user’s interactions, it enters clarification dialogs to revise its user model and to adapt the explanation. The architecture of the dialog planner can also be used to adapt content selection and explicitness reactively to the audience’s needs. The rationale behind the architecture should prove to be useful for explanation systems in general.

However, in the current experimental stage, only a small set of user interactions is allowed. More elaborate interactions

that call for more complex reactions are desirable. Therefore, empirical studies of teacher-student interactions in mathematics classes are necessary.

References

- [Anderson and Lebiere, 1998] John R. Anderson and Christian Lebiere. *The Atomic Components of Thought*. Lawrence Erlbaum, 1998.
- [Cahill *et al.*, 1999] Lynne Cahill, Christy Doran, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, Donia Scott, and Neil Tipper. In search of a reference architecture for NLG systems. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 77–85, Toulouse, France, 1999.
- [Cawsey, 1990] Alison Cawsey. Generating explanatory discourse. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, number 4 in Cognitive Science Series, pages 75–101. Academic Press, San Diego, CA, 1990.
- [Horacek, 1997] Helmut Horacek. A model for adapting explanations to the user’s likely inferences. *User Modeling and User-Adapted Interaction*, 7:1–55, 1997.
- [Hovy, 1993] Eduard H. Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63:341–385, 1993.
- [Huang and Fiedler, 1997] Xiaorong Huang and Armin Fiedler. Proof verbalization as an application of NLG. In Martha E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 965–970, Nagoya, Japan, 1997. Morgan Kaufmann.
- [Huang, 1994] Xiaorong Huang. *Human Oriented Proof Presentation: A Reconstructive Approach*. PhD thesis, Fachbereich Informatik, Universität des Saarlandes, Saarbrücken, Germany, 1994.
- [Kilger and Finkler, 1995] Anne Kilger and Wolfgang Finkler. Incremental generation for real-time applications. Research Report RR-95-11, DFKI, Saarbrücken, Germany, July 1995.
- [Mann and Thompson, 1987] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. ISI Reprint Series ISI/RS-87-190, University of Southern California/Information Science Institute, Marina del Rey, CA, 1987.
- [Mooney *et al.*, 1991] David J. Mooney, Sandra Carberry, and Kathleen McCoy. Capturing high-level structure of naturally occurring, extended explanations using bottom-up strategies. *Computational Intelligence*, 7:334–356, 1991.
- [Moore and Swartout, 1991] Johanna D. Moore and William R. Swartout. A reactive approach to explanation: Taking the user’s feedback into account. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence*, pages 3–48, Boston, MA, USA, 1991. Kluwer.
- [Paris, 1991] Cécile Paris. The role of the user’s domain knowledge in generation. *Computational Intelligence*, 7:71–93, 1991.
- [Reiter, 1994] Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport, Maine, USA, 1994.
- [Wahlster *et al.*, 1993] Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, and Thomas Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence*, 63:387–427, 1993.

Generating Tailored Examples to Support Learning via Self-explanation

Cristina Conati and Giuseppe Carenini

Department of Computer Science
University of British Columbia
Vancouver, BC, Canada, V6T 1Z4
{conati, carenini}@cs.ubc.ca

Abstract

We describe a framework that helps students learn from examples by generating example problem solutions whose level of detail is tailored to the students' domain knowledge. The framework uses natural language generation techniques and a probabilistic student model to selectively introduce gaps in the example solution, so that the student can practice applying rules learned from previous examples in problem solving episodes of difficulty adequate to her knowledge. Filling in solution gaps is part of the meta-cognitive skill known as self-explanation (generate explanations to oneself to clarify an example solution), which is crucial to effectively learn from examples. In this paper, we describe how examples with tailored solution gaps are generated and how they are used to support students in learning through gap-filling self-explanation.

1 Introduction

Studying examples is one of the most natural ways of learning a new skill. Thus, substantial research in the field of Intelligent Tutoring Systems (ITS) has been devoted to understand how to use examples to enhance learning. Most of this research has focused on how to select examples that can help a student during problem solving [e.g., Burrow and Weber 1996; Aleven and Ashley 1997]. In this paper, we focus on how to describe an example solution so that a student can learn the most by studying it previous to problem solving. In particular, we address the issue of how to vary the level of detail of the presented example solution, so that the same example can be equally stimulating for learners with different degrees of domain knowledge.

This problem is novel in ITS, as it requires sophisticated natural language generation (NLG) techniques. While the NLG field has extensively studied the process of producing text tailored to a model of the user's inferential capabilities [e.g., Horacek 1997; Korb, McConachy et al. 1997; Young 1999], the application of NLG techniques in ITS are few and mainly focused on managing and structuring the tutorial dialogue [e.g., Moore 1996; Freedman 2000], rather than on tailoring the presentation of instructional material to a detailed student model.

The rationale behind varying the level of detail of an example solution lies on cognitive science studies showing that those students who self-explain examples (i.e., generate explanations to themselves to clarify an example solution) learn better than those students who read the examples without elaborating them [Chi 2000]. One kind of self-explanation that these studies showed to be correlated with learning involves filling in the gaps commonly found in textbook example solutions (*gap filling* self-explanation). However, the same studies also showed that most students tend not to self-explain spontaneously. In the case of gap filling, this phenomenon could be due to the fact that gap filling virtually requires performing problem solving steps while studying an example. And, because problem solving can be highly cognitively and motivationally demanding [Sweller 1988], if the gaps in an example solution are too many or too difficult for a given student, they may hinder self-explanations aimed at filling them.

We argue that, by monitoring how a student's knowledge changes when studying a sequence of examples, it is possible to introduce in the examples solution gaps that are not too cognitively demanding, thus facilitating gap filling self-explanation and providing a smooth transition from example study to problem solving. We are testing our hypothesis by extending the SE-Coach, a framework to support self-explanation of physics examples [Conati and VanLehn 2000].

The SE-Coach already effectively guides two other kinds of self-explanations that have been shown to trigger learning [Chi 2000]: (i) justify a solution step in terms of the domain theory (*step correctness*); (ii) map a solution step into the high-level plan underlying the example solution (*step utility*). The internal representation of an example solution used by the SE-Coach to monitor students' self-explanation is generated automatically. However, because the SE-Coach does not include any NLG capability, the example description presented to the student and the mapping between this description and the internal representation is done by hand. Thus, each example has a fixed description, containing virtually no solution gaps.

In this paper, we describe how we extended the SE-Coach with NLG techniques to (i) automatically generate the

example presentation from the example internal representation (ii) selectively insert gaps in the example presentation, tailored to a student's domain knowledge.

Several NLG computational models proposed in the literature generate concise text by taking into account the inferential capabilities of the user. [Young 1999] generates effective plan descriptions tailored to the hearer's *plan reasoning* capabilities. [Horacek 1997] is an example of models that take into account the hearer's *logical inference* capabilities. And [Korb, McConachy et al. 1997] proposes a system that relies on a model of user's *probabilistic inferences* to generate sufficiently persuasive arguments.

In contrast, our generation system tailors the content and organisation of an example to a *probabilistic model of the user logical inferences*, which allows us to explicitly represent the inherent uncertainty involved in assessing a learner's knowledge and reasoning processes. Furthermore, our system maintains information on what example parts are not initially presented (i.e., solution gaps), which is critical to support gap-filling self-explanations for those students who tend not to self-explain autonomously.

In the following sections, we first illustrate our general framework for example generation. We then describe in detail the NLG techniques used and an example of the tailored presentations they generate. Finally, we show how the output of the NLG process supports an interface to guide gap filling self-explanation.

2 The Framework for Example Generation

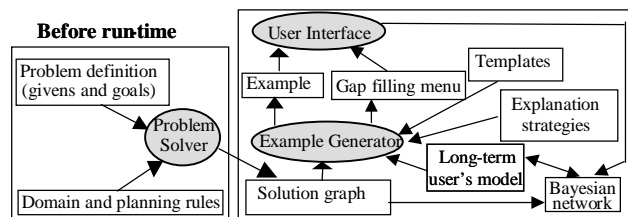


Figure 1: Framework for example generation

Figure 1 shows the architecture of our framework for generating tailored example presentations. The part of the framework labelled “before run-time” is responsible for generating the internal representation of an example solution from (i) a knowledge base (KB) of domain and planning rules (for physics in this particular application); (ii) a formal description of the example initial situation, given quantities and sought quantities [Conati and VanLehn 2000]. A problem solver uses these two knowledge sources to generate the example solution represented as a dependency network, known as the *solution graph*. The solution graph encodes how each intermediate result in the example solution is derived from a domain or planning rule and from previous results matching that rule's preconditions. Consider, for instance, the physics example in Figure 2 (Example1). Figure 3 shows the part of solution graph that derives the first three steps mentioned in Example1 solution:

establish the goal to apply Newton's 2nd Law; select the body to which to apply the law; identify the existence of a tension force on the body.

In the solution graph, intermediate solution facts and goals (F- and G- nodes in Figure 3) are connected to the rules (R-nodes) used to derive them and to previous facts and goals matching these rules' enabling conditions. The connection goes through rule-application nodes (RA- nodes in Figure 3), explicitly representing the application of each rule in the context of a specific example. Thus, the segment of network in Figure 3 encodes that the rule *R-try-Newton-2law* establishes the goal to apply Newton's 2nd Law (node *G-try-Newton-2law*) to solve the goal to find the force on Jake (node *G-force-on Jake*).

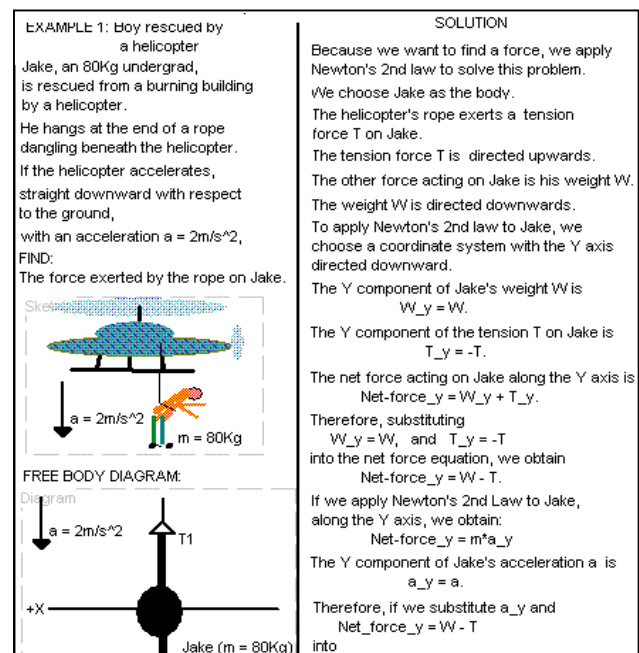


Figure 2: Sample Newtonian physics example

The rule *R-goal-choose-body* sets the subgoal to find a body to apply the Newton's 2nd Law (node *G-goal-choose-body*), while the rule *R-find-forces* sets the subgoal to find all the forces on the body (node *G-find-forces*). The rule *R-body-by-force* dictates that, if one has the goals to find the force on an object and to select a body to apply Newton's 2nd Law, that object should be selected as the body. Thus, in Figure 3 this rule selects Jake as the body for Example1 (node *F-Jake-is the body*). The rule *R-tension-exists* says that if an object is tied to a taut string, then there is a tension force exerted by the string on the object. When applied to Example1, this rule generates the fact that there is a tension force on Jake (node *F-tension-on-Jake* in Figure 3).

The solution graph can be seen as a model of correct self-explanation for the example solution, because for each solution fact it encodes the various types of self-explanations relevant to understand it: *step correctness* (what domain rule generated that fact), *step utility* (what

goal that fact fulfils) and *gap filling* (how the fact derives from previous solution steps).

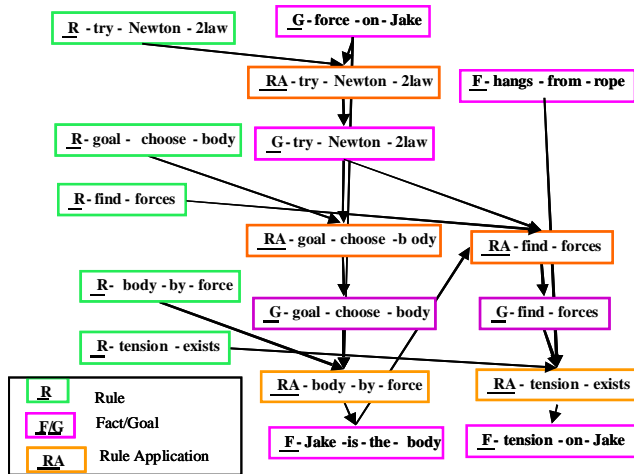


Figure 3: Segment of solution graph for Example 1

In the SE-Coach, every time a student is shown an example, the corresponding solution graph provides the structure for a Bayesian network (see right bottom side of Figure 1) that uses information on how the student reads and self-explains that example to generate a probabilistic assessment of how well the student understands the example and the related rules [Conati and VanLehn 2001]. The prior probabilities to initialise the rule nodes in the Bayesian network come from the long-term student model (see Figure 1), which contains a probabilistic assessment of a student's current knowledge of each rule in the KB. This assessment is updated every time the student finishes studying an example, with the new rule probabilities computed by the corresponding Bayesian network.

In the SE-Coach, the solution graph and Bayesian network described above are used to support students in generating self-explanations for correctness and utility only. No explicit monitoring and support for gap filling self-explanation is provided. This is because in the SE-Coach, the description of the example solutions presented to the student and the mapping between these descriptions and the corresponding solution graphs are done by hand. This makes it impossible to tailor an example description to the dynamically changing student model by inserting gaps at the appropriate difficulty level for a given student. We have overcome this limitation by adding to the SE-Coach the example generator (see right part of Figure 1), a NLG system that can automatically tailor the detail level of an example description to the student's knowledge, in order to stimulate and support gap-filling self-explanation.

3 The Example Generator (EG)

EG is designed as a standard pipelined NLG system [Reiter and Dale 2000]. A text planner [Young and Moore 1994] selects and organizes the example content, then a

microplanner and a sentence generator realize this content into language. In generating an example, EG relies on two key communicative knowledge sources (right part of Figure 1): (i) a set of explanation strategies that allow the text planner to determine the example's content, organization and rhetorical structure; (ii) a set of templates that specifies how the selected content can be phrased in English.

The design of these sources involved a complex acquisition process. We obtained an abstract model of an example's content and organisation from a detailed analysis of the rules used to generate the solution graph. This was combined with an extensive examination of several physics textbook examples, which also allowed us to model the examples' rhetorical structure and the syntactic and semantic structure of their clauses. To analyse the rhetorical structure of the examples, we followed Relational Discourse Analysis (RDA) [Moser, Moore et al. 1996], a coding scheme devised to analyse tutorial explanations. The semantic and syntactic structure of the examples' clauses was used to design the set of templates that map content into English.

We now provide the details of the selection and organisation of the example content. In EG, this process relies on the solution graph and on the probabilistic long term student model. It consists of two phases, text planning and revision, to reduce the complexity of the plan operators and increase the efficiency of the planning process. Text planning selects from the solution graph a knowledge pool of all the propositions (i.e., goals and facts) necessary to solve a given example, and it organizes them according to ordering constraints also extracted from the solution graph. The output of this phase, if realized, would generate a fully detailed example solution. After text planning, a revision process uses the assessment in the student's long-term model to decide whether further content selection can be performed to insert appropriate solution gaps. Text planning and revision are described in the following sub-sections.

3.1 Text Planning Process

The input to the text planner consists of (i) the abstract communicative action of describing an example solution; (ii) the example solution graph; (iii) the explanation strategies. The planning process selects and organizes the content of the example solution by iterating through a loop of communicative action decomposition¹. Abstract actions are decomposed until primitive communicative actions (executable as speech acts) are reached. In performing this task, the text planner relies on the set of explanation strategies that specify possible decompositions for each communicative action and the constraints dictating when they may be applied. These constraints are checked against

¹ Communicative actions satisfy communicative goals. So, text planning actually involves two intertwined processes of goal and action decomposition. To simplify our presentation, we only refer to communicative actions and their decomposition.

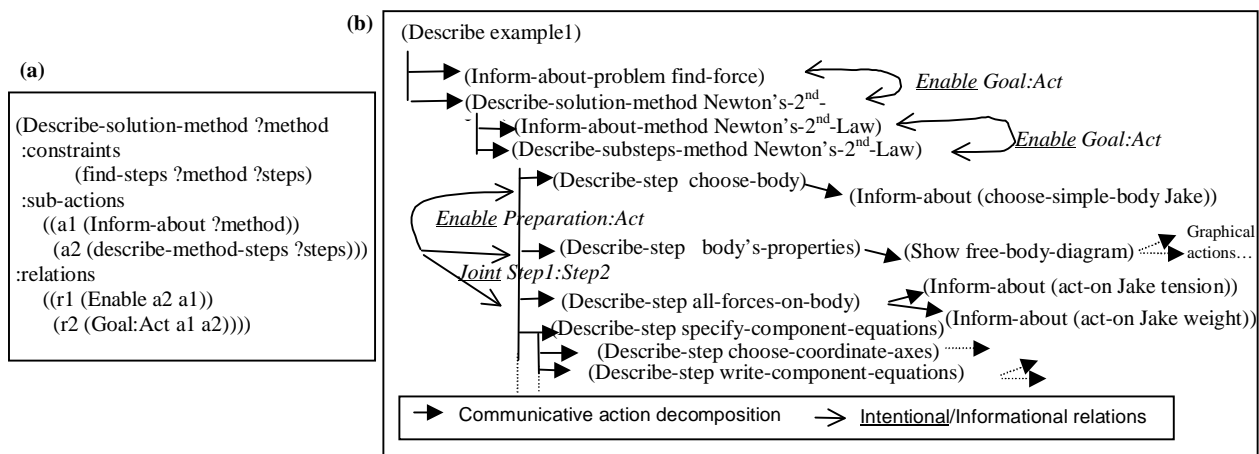


Figure 4: (a) Sample explanation strategy. (b) Portion of the text plan

the solution graph and when they are satisfied the decomposition is selected and appropriate content is also extracted from the solution graph. For illustration, Figure 4(a) shows a simplified explanation strategy that decomposes the communicative action *describe-solution-method*. Possible arguments for this action are, for instance, the Newton's-2nd-Law and the Conservation-of-Energy methods. Looking at the details of the strategy, the function *find-steps* (:constraints field) checks in the solution graph whether the method has any steps. If this is the case, the steps are retrieved from the solution graph and the *describe-solution-method* action is decomposed in an *inform-about* primitive action and in a *describe-method-steps* abstract action. The output of the planning process is a text plan, a data structure that specifies what propositions the example should convey, a partial order over those propositions and the example rhetorical structure. A portion of the text plan generated by EG for Example1 is shown in Figure 4(b).

The propositions that the example should convey are specified as arguments of the primitive actions in the text plan. In Figure 4(b) all primitive actions are of type *inform*. For instance, the primitive action (*Inform-about (act-on Jake weight)*) specifies the proposition (*act-on Jake weight*), which is realized in the example description as “the other force acting on Jake is his weight”. In the text plan, the communicative actions are partially ordered. This ordering is not shown in the figure for clarity's sake; the reader can assume that the actions are ordered starting at the top. The example rhetorical structure consists of the action decomposition tree and the informational/intentional relations among the communicative actions. For instance, in (b), the rhetorical structure associated with the action *describe-solution-method* specifies that, to describe the solution method, the system has to perform two actions: (i) *inform the user about the method adopted*; (ii) *describe all the steps of the method*. Between these two actions the *Enable* intentional relation and the *Goal:Act* informational relation hold. All the informational /intentional relations used in EG are discussed in [Moser, Moore et al. 1996]. We

clarify here only the meaning of the *Enable* relation because this relation is critical in supporting gap-filling self-explanations. An intentional *Enable* relation holds between two communicative actions if one provides information intended to increase either the hearer's understanding of the material presented by the other, or her ability to perform the domain action presented by the other.

3.2 The Revision Process

Once the text planner has generated a text plan for the complete example, the revision process revises the plan to possibly insert solution gaps that can make the example more stimulating for a specific student. The idea is to insert solution gaps of adequate difficulty, so that the student can practice applying newly acquired knowledge without incurring in the excessive cognitive load that too demanding problem solving can generate [Sweller 1988].

The revision process performs further content selection by consulting the probabilistic long-term student model that estimates the current student's domain knowledge. More specifically, the revision process examines each proposition specified by a primitive communicative action in the text plan and, if according to the student model, there is high probability that the student knows the rule necessary to infer that proposition, the action is de-activated. De-activated actions are kept in the text plan but are not realized in the text, thus creating solution gaps. However, as we will see in the next section, de-activated actions may be realized in follow-up interactions.

As an illustration of the effects of the revision process on content selection, compare the example solutions shown in Figure 6 and Figure 7. Figure 6 displays the worked out solution for Example2 which, similarly to Example1, does not contain any solution gaps. In contrast, the same portion of Example2 solution shown in Figure 7 is much shorter, including several solution gaps. As previously described, EG determines what information to leave out by consulting the long-term probabilistic student model. In particular, the concise solution in Figure 7 is generated by EG if the

student had previously studied Example1 with the SE-Coach and generated self-explanations of correctness and utility providing sufficient evidence that she understands the rules used to derive Example1 solution. When selecting the content for Example2, EG leaves out all the propositions derived from the rules that the student has learned from Example1. Notice, for instance, that the concise solution in Figure 7 does not mention the solution method used and the weight force. Also, the choice of the body and of the coordinate system is only conveyed indirectly.

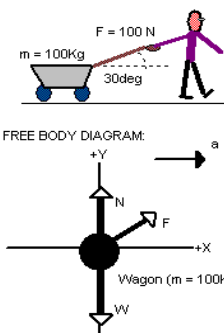
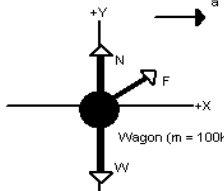
EXAMPLE 2: Person pulling a wagon	SOLUTION
<p>A person pulls a loaded wagon. The wagon has mass $m = 100\text{Kg}$. The person pulls it with a force F of 100 Newtons, applied at 30 degrees from the horizontal.</p> <p>FIND</p> <p>1) the force N exerted on the wagon by the ground</p> <p>2) The acceleration a of the wagon.</p>  <p>FREE BODY DIAGRAM:</p> 	<p>We solve this problem by applying Newton's 2nd law. We choose the wagon as the body.</p> <p>One of the forces acting on the wagon is its weight W. The wagon's weight W is directed downward.</p> <p>The force N exerted by the ground on the wagon is a normal force. This normal force N is directed upward.</p> <p>We choose a coordinate system with the X axis directed to the right and the Y axis directed upward. Therefore the weight has components:</p> $W_y = -W$ $W_x = 0.$ <p>The normal force N has components:</p> $N_y = N$ $N_x = 0.$ <p>Finally, the pulling force F on the wagon has components:</p> $F_y = F \cdot \cos(60)$ $F_x = F \cdot \cos(30)$ <p>Because the Y component of the net force on the wagon is:</p> $\text{Net-force}_y = N_y + W_y + F_y,$ <p>the Y component of the wagon's acceleration is:</p> $a_y = 0$ <p>the general equation for Newton's 2nd law applied to the wagon along the Y axis, $\text{Net-Force}_y = m \cdot a_y$, becomes:</p> $N - W + F \cdot \cos(60) = 0.$ <p>Since the value of the wagon's weight W is:</p> $W = m \cdot g = 980 \text{ Newtons},$

Figure 6 Portion of Example2 without solution gaps

Even if a student has sufficient knowledge to fill in the solution gaps inserted by the revision process, she may not actually perform the required inferences when studying the example. As a matter of fact, cognitive science studies show that most students tend not to self-explain spontaneously [Chi 2000]. Thus, once the text plan is revised and realized, the system presents the concise example with tools designed to stimulate gap filling self-explanation as we illustrate in the next section.

4 Support for Gap Filling Self-explanation

To support gap-filling self-explanation, we have extended the interface that the SE-Coach uses to support self-explanations for step correctness and utility. In this interface, each example's graphical element and solution step presented to the student is covered with grey boxes. Figure 8(a) shows a segment of the example solution in Figure 7 as presented with the masking interface.

To view an example part, the student must move the mouse over the box that covers it, thus allowing the interface to track what the student is reading. When the

SOLUTION
<p>The force N exerted by the ground on the wagon is a normal force.</p> <p>This normal force N is directed upward.</p> <p>The pulling force F on the wagon has components:</p> $F_y = F \cdot \cos(60)$ $F_x = F \cdot \cos(30)$ <p>and the Y component of the wagon's acceleration is:</p> $a_y = 0$ <p>The general equation for Newton's 2nd law applied to the wagon along the Y axis, $\text{Net-Force}_y = m \cdot a_y$, becomes:</p> $N - W + F \cdot \cos(60) = 0.$ <p>Since the value of the wagon's weight W is:</p> $W = m \cdot g = 980 \text{ Newtons},$

Figure 7 Portion of Example2 with solution gap

EXAMPLE 2:	SOLUTION	(a)
<p>The force N exerted on the wagon by the ground is a normal force</p>	<p>The force N exerted on the wagon by the ground is a normal force</p>	<p>Self-Explain</p> <p>Filling in missing step(s)</p> <p>This fact is true because...</p> <p>The role of this fact in the solution plan is...</p>
<p>Text item for gap</p> <p>The force N exerted on the wagon by the ground is a normal force</p>	<p>Text item for gap</p> <p>The force N exerted on the wagon by the ground is a normal force</p>	<p>FILLING MISSING STEPS</p> <p>Fill in the following missing step(s)</p> <p>Text item for gap</p> <p>Submit Done</p>

Figure 8 Interface tools for gap filling self-explanation

student uncovers an example part, a "self-explain" button appears next to it (see Figure 8(a)). Clicking on this button generates more specific prompts that suggest one or more of the self-explanations for correctness, utility or gap filling, depending upon which of them are needed by the current student to fully understand the uncovered step. In particular, the text plan produced by EG is the key element in determining whether a prompt for gap filling is generated. A prompt for gap filling is generated whenever some of the primitive communicative actions that were de-activated during the revision process are related through an *Enable* intentional relation to the communicative action expressing the uncovered example part. The rationale behind this condition is that a solution gap with respect to an example part comprises all the solution steps that were left out, but whose understanding is a direct precondition to derive that example part. For instance, given the example part uncovered in Figure 8(a), there is only one solution gap preceding it, namely the one corresponding to the communicative action *Inform-about (choose-simple-body-Jake)*². As shown in Figure 8(a), the prompt for gap filling is generated by adding the item "filling in missing steps" to the self-explain menu. If the student clicks on this item, the

² Since the text plans for Example1 and Example2 are structurally the same, this can be verified in Figure 4(b)

interface inserts in the solution text an appropriate number of masking boxes, representing the missing steps (see Figure 8(b), left panel, first box from top). The interface also activates a dialogue box containing a blank for each missing step, that the student can use to fill in the step (see Figure 8(b), right panel). Since the interface currently does not process natural language input, the student fills each blank by selecting an item in the associated pull-down menu. EG generates the entries in this menu by applying the realisation component to unrealised communicative actions in the text plan (see Figure 1).

The student receives immediate feedback on the correctness of his selection, which is also sent to the Bayesian network built for the current example (see Figure 1). The network fact node that corresponds to the missing step is clamped to either true or false, depending on the correctness of the student's selection, and the network updates the probability of the corresponding rule consequently. Thus, if the student's actions show that he is not ready to apply a given rule to fill a solution gap, this rule's probability will decrease in the long-term student model. As a consequence, the next presented example involving this rule will include the solution steps the rule generates, giving the student another opportunity to see how the rule is applied.

5 Conclusions and Future Work

We have presented a tutoring framework that integrates principles and techniques from ITS and NLG to improve the effectiveness of example studying for learning. Our framework uses an NLG module and a probabilistic student model to introduce solution gaps in the example solutions presented to a student. Gaps are introduced when the student model assesses that the student has gained from previous examples sufficient knowledge of the rules necessary to derive the eliminated steps. The goal is to allow the student to practice applying these rules in problem solving episodes of difficulty adequate for his knowledge.

Our framework is innovative in two ways. First, it extends ITS research on supporting the acquisition of the learning skill known as self-explanation, by providing tailored guidance for gap filling self-explanation. Second, it extends NLG techniques on producing user-tailored text by relying on a dynamically updated probabilistic model of the user logical inferences.

The next step in our research will be to test the effectiveness of our framework through empirical studies. These studies are crucial to refine the probability threshold currently used to decide when to leave out a solution step, and possibly to identify additional principles to inform the text plan revision. Additional future work involves NLG research on how the example text plan can be used to maintain the coherence of the other example portions, when the student fills a solution gap.

References

- [Aleven and Ashley 1997] Aleven, V. and K. Ashley. Teaching case-based argumentation through a model and examples: empirical evaluation of an intelligent learning environment. *AIED'99*, Kobe, Japan, August 1997.
- [Burrow and Weber 1996] Burrow, R. and G. Weber. Example explanation in learning environments. *Intelligent Tutoring Systems - Proceedings of the Third International Conference, ITS '96*, Springer, June 1996.
- [Chi 2000] Chi, M. T. H. Self-Explaining Expository Texts: The Dual Processes of Generating Inferences and Repairing Mental Models. *Advances in Instructional Psychology*. R. Glaser. Mahwah, NJ, Lawrence Erlbaum Associates: 161-238, 2000.
- [Conati and VanLehn 2001] Conati, C. and K. VanLehn. "Providing adaptive support to the understanding of instructional material". *Proc. of IUI 2001, International Conference on Intelligent User Interfaces*, Santa Fe, NM, January 2001.
- [Conati and VanLehn 2000] Conati, C. and K. VanLehn. "Toward Computer-Based Support of Meta-Cognitive Skills: a Computational Framework to Coach Self-Explanation." *Int. Journal of AI in Education* 11, 2000.
- [Freedman 2000] Freedman, R. Plan-based dialogue management in a physics tutor. *Sixth Applied Natural Language Processing Conference*, Seattle, WA, 2000.
- [Horacek 1997] Horacek, H. "A Model for Adapting Explanations to the User's Likely Inferences." *UMUAI* 7(1): 1-55, 1997.
- [Korb, McConachy et al. 1997] Korb, K. B., R. McConachy, et al. A Cognitive Model of Argumentation. *Proc. 19th Cognitive Science Conf*, Stanford, CA, August 1997.
- [Moore 1996] Moore, J. "Discourse generation for instructional applications: Making computer-based tutors more like humans." *Journal of AI in Education* 7(2), 1996.
- [Moser, Moore et al. 1996] Moser, M. G., J. D. Moore and E. Glendeling. Instructions for Coding Explanations: Identifying Segments, Relations and Minimal Units, TR 96-17 Univ. of Pittsburgh, Dept. of Computer Science, 1996.
- [Reiter and Dale 2000] Reiter, E. and R. Dale. *Building NLG Systems*, Cambridge University Press, 2000.
- [Sweller 1988] Sweller, J. "Cognitive load during problem solving: effects on learning." *Cognitive Science* 12: 257-285, 1988.
- [Young 1999] Young, M. "Using Grice's maxim of Quantity to select the content of plan descriptions." *Artificial Intelligence Journal* 115(2): 215-256, 1999.
- [Young and Moore 1994] Young, M. and J. Moore. DPOCL: A Principled Approach to Discourse Planning. *Proc. 7th Int. Workshop on NLG*, 13-20, 1994.

An Empirical Study of the Influence of User Tailoring on Evaluative Argument Effectiveness

Giuseppe Carenini

Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada V6T 1Z4
carenini@cs.ubc.ca

Johanna D. Moore

The Human Communication Research Centre,
University of Edinburgh,
2 Buccleuch Place, Edinburgh EH8 9LW, UK.
jmoore@cogsci.ed.ac.uk

Abstract

The ability to generate effective evaluative arguments is critical for systems intended to advise and persuade their users. We have developed a system that generates evaluative arguments that are tailored to the user, properly arranged and concise. We have also devised an evaluation framework in which the effectiveness of evaluative arguments can be measured with real users. This paper presents the results of a formal experiment we performed in our framework to verify the influence of user tailoring on argument effectiveness.

1 Introduction

Evaluative arguments are pervasive in natural human communication. In countless situations, people attempt to advise or persuade their interlocutors that something is good (vs. bad) or right (vs. wrong). For instance, doctors need to advise their patients on which treatment is best for them (the patients). A teacher may need to convince a student that a certain course is (is not) the best choice for the student. And a sales person may need to compare two similar products and argue why her current customer should like one more than the other. With the explosion of the information available on-line and the ever-increasing availability of wireless devices, we are witnessing a proliferation of computer systems that aim to support or replace humans in similar communicative settings. Clearly, the success of these systems serving as personal assistants, advisors, or shopping assistants (e.g., [Chai, Budzikovska et al. 2000]) may crucially depend on their ability to generate and present effective evaluative arguments.

In the last decade, considerable research has been devoted to develop computational models for automatically generating and presenting evaluative arguments. Several studies investigated the process of selecting and structuring the content of the argument (e.g., [Morik 1989; Elzer, Chu-Carroll et al. 1994; Klein 1994]), while [Elhadad, McKeown et al. 1997] developed a detailed model of how the selected content should be realised into natural language. All these approaches to evaluative argument generation follow a basic guideline from argumentation theory [Mayberry and Golden 1996]: effective evaluative arguments should be constructed

considering the values and preferences of the audience towards the information presented. In practice, this means that all previous approaches tailor the generated arguments to a model of the user's values and preferences.

However, a key limitation of previous work is that none of the proposed approaches has been empirically evaluated. Thus, in particular, it is not clear whether and to what extent tailoring an evaluative argument to a model of the user increases its effectiveness. The work presented in this paper is a first step toward addressing this limitation. By recognising the fundamental role of empirical testing in assessing progress, generating new research questions and stimulating the acceptance of a technique as viable technology, we have performed an experiment to test the influence of user-tailoring on argument effectiveness. In the remainder of the paper, we first provide a short description of our system for generating evaluative arguments tailored to a model of the user's preferences. Then, we briefly present a framework to measure the effectiveness of evaluative arguments with real users. Next, we discuss the experiment we ran within the framework to test the influence of user tailoring on evaluative argument effectiveness.

2 User Tailored Evaluative Arguments

Our generation system, known as the Generator of Evaluative Argument (GEA) [Carenini 2000], generates evaluative arguments whose content, organisation and phrasing are tailored to a quantitative model of the user's values and preferences. The model is expressed as an Additive Multiattribute Value Function (AMVF), a conceptualization based on MultiAttribute Utility Theory (MAUT) [Clemen 1996]. Besides being widely used in decision theory (where they were originally developed), conceptualizations based on MAUT have recently become a common choice in the user modeling field [Jameson, Schafer et al. 1995]. Furthermore, similar models are also used in Psychology, in the study of consumer behaviour [Solomon 1998]. In GEA, a user specific AMVF is a key knowledge source in all the phases of the generation process. GEA is implemented as a standard

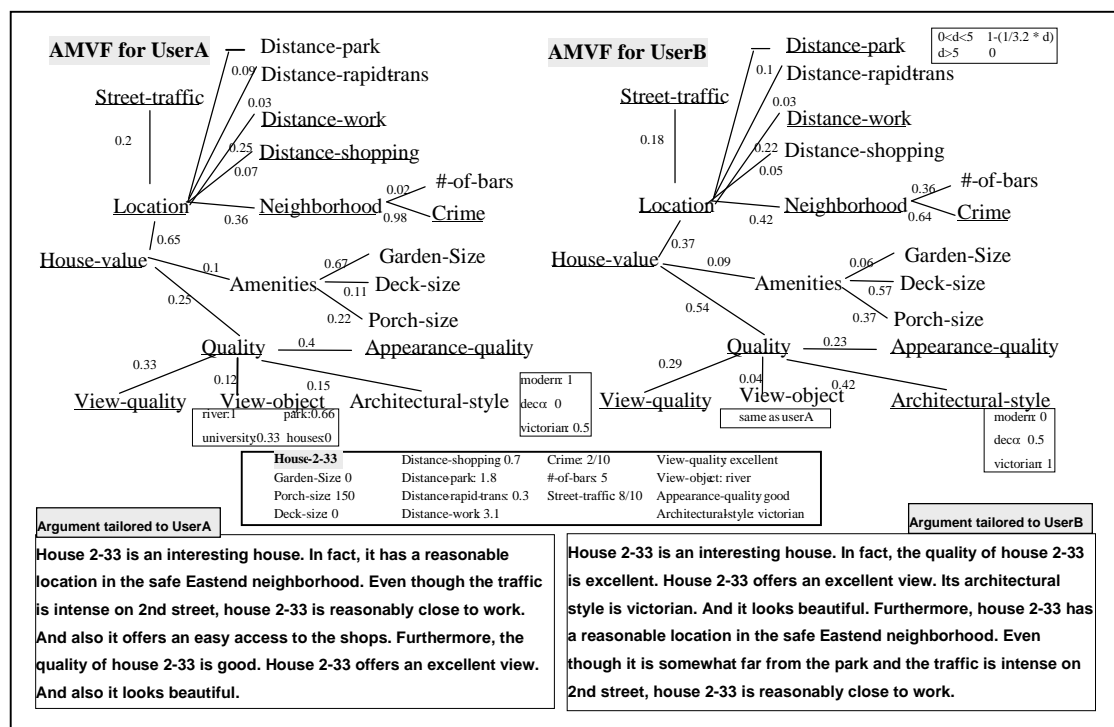


Figure 1 Top: AMVF for two sample users; for clarity's sake only a few component value functions are shown. Bottom: arguments about house-2-33 tailored to the two different models

pipelined generation system, including a text planner, a microplanner, and a sentence realizer.

2.1 AMVFs and their Use in GEA

An AMVF is a model of a person's values and preferences with respect to entities in a certain class. It comprises a *value tree* and a set of *component value functions*. A value tree is a decomposition of an entity value into a hierarchy of entity aspects (called objectives in decision theory), in which the leaves correspond to the entity primitive objectives (see top of Figure 1 for two simple value trees in the real estate domain). The arcs in the tree are weighted to represent the importance of an objective with respect to its siblings (e.g., in Figure 1 *location* for UserA is more than twice as important as *quality* in determining the *house-value*). The sum of the weights at each level is always equal to 1. A component value function for a primitive objective expresses the preferability of each value for that objective as a number in the [0,1] interval, with the most preferable value mapped to 1, and the least preferable one to 0. For instance, in Figure 1 the *victorian* value of the primitive objective *architectural-style* is the most preferred by UserB, and a *distance-from-park* of 1 mile has for UserB preferability $(1 - (1/3.2 * 1))=0.69$. Formally, an AMVF predicts the value $v(e)$ of an entity e as follows:

$$v(e) = v(x_1, \dots, x_n) = \sum w_i v_i(x_i), \text{ where}$$

- (x_1, \dots, x_n) is the vector of primitive objective values for an entity e

- \forall primitive objective i , v_i is the component value function and w_i is its weight, with $0 \leq w_i \leq 1$ and $\sum w_i = 1$; w_i is equal to the product of all the weights on the path from the *root* of the value tree to the primitive objective i .

Thus, given someone's AMVF, it is possible to compute how valuable an entity is to that individual. Although for lack of space we cannot provide details here, given a user specific AMVF and an entity, GEA can also compute additional precise measures that are critical in generating a user-tailored evaluative argument for that entity. First, GEA can compute how valuable any objective of the entity is for that user. This information plays an essential role in phrasing the argument by determining the selection of scalar adjectives (e.g., convenient), which are the basic linguistic resources to express evaluations. Second, GEA can identify what objectives can be used as supporting or opposing evidence for an evaluative claim. Third, GEA can compute for each objective the strength of supporting (or opposing) evidence it can provide in determining the evaluation of its parent objective. In this way, in compliance with argumentation theory, evidence can be arranged according to its strength and concise arguments can be generated by only including sufficiently strong evidence [Carenini and Moore 2000]. The measure of evidence strength and the threshold that defines when a piece of evidence is worth mentioning were adapted from [Klein 1994].

A final note on AMVF's applicability. According to decision theory, in the general case, when uncertainty is

present, user's preferences for an entity can be represented as an AMVF only if her preferences for the primitive objectives satisfy a stringent condition (i.e., additive independence). However, evidence has shown that an AMVF is a reasonable model of most people's preferences under conditions of certainty [Clemen 1996]. We felt that we could safely use AMVFs in our study, because we selected the objectives to avoid possible violations of additive independence. And we considered a situation with no uncertainty.

2.2 An Example: Generating Arguments for Two Different Users

Figure 1 illustrates how the content, organization and phrasing of the arguments generated by GEA are sensitive to the model of a user's preferences. The top of the figure shows two different models of actual users in the real-estate domain. The bottom of the figure shows two evaluative arguments generated for the same house but tailored to the two different models. The primitive objectives' values for the house are reported in the middle of the figure. Notice how the two arguments differ substantially. Different objectives are included (the objectives included are underlined in the two models). Furthermore, the objectives are ordered differently (e.g., in the first argument *location* comes before *quality*, whereas the opposite is true in the second argument). Finally, the evaluations are also different. For instance, *quality* is *good* for UserA, but *excellent* for UserB.

3 The Evaluation Framework

To run our formal experiment, we used an evaluation framework based on the *task efficacy* evaluation method [Carenini 2000]. This method allows the experimenter to evaluate a generation model indirectly, by measuring the effects of its output on user's behaviors, beliefs and attitudes in the context of a task. Aiming at general results, we chose a basic and frequent task that has been extensively studied in decision analysis: the selection of a subset of preferred objects (e.g., houses) out of a set of possible alternatives. In our evaluation framework, the user performs this task by using a system for interactive data exploration and analysis (IDEA), see Figure 3. Let's now examine how GEA can be evaluated in the context of the selection task, by going through the evaluation framework architecture.

3.1 The Evaluation Framework Architecture

As shown in Figure 2, the evaluation framework consists of four main sub-systems: the IDEA system, the User Model Refiner, the New Instance Generator and GEA. The framework assumes that a model of the user's preferences (an AMVF) has been previously acquired from the user, to assure a reliable initial model. At the onset, the user is assigned the task to select from the dataset the four most

preferred alternatives and to place them in a Hot List (see Figure 3, upper right corner) ordered by preference. Whenever the user feels that the task is accomplished, the ordered list of preferred alternatives is saved as her Preliminary Hot List (Figure 2 (2)). After that, this list and the initial Model of User's Preferences are analysed by the User Model Refiner to produce a Refined Model of the User's Preferences (Figure 2 (3)). Then a New Instance (NewI) is designed on the fly by the New Instance Generator to be preferable for the user given her refined preference model (Figure 2 (4)). At this point, the stage is set for argument generation. Given the Refined Model of the User's Preferences, the Argument Generator produces an evaluative argument about NewI tailored to the model (Figure 2 (5)), which is presented to the user by the IDEA system (Figure 2 (6))(see also Figure 3 for an example). The argument goal is to persuade the user that NewI is worth being considered. Notice that all the information about NewI is also presented graphically.

Once the argument is presented, the user may (a) decide immediately to introduce NewI in her Hot List, or (b) decide to further explore the dataset, possibly making changes and adding NewI to the Hot List, or (c) do nothing.

Figure 3 shows the display at the end of the interaction, when the user, after reading the argument, has decided to introduce NewI in the Hot List first position (Figure 3, top right).

Whenever the user decides to stop exploring and is satisfied with her final selection, measures related to argument's effectiveness can be assessed (Figure 2 (7)). These measures are obtained either from the record of the user interaction with the system or from user self-reports in a final questionnaire (see Figure 4 for an example of self-report) and include:

- Measures of behavioral intentions and attitude change: (a) whether or not the user adopts NewI, (b) in which position in the Hot List she places it and (c) how much she likes NewI and the other objects in the Hot List.

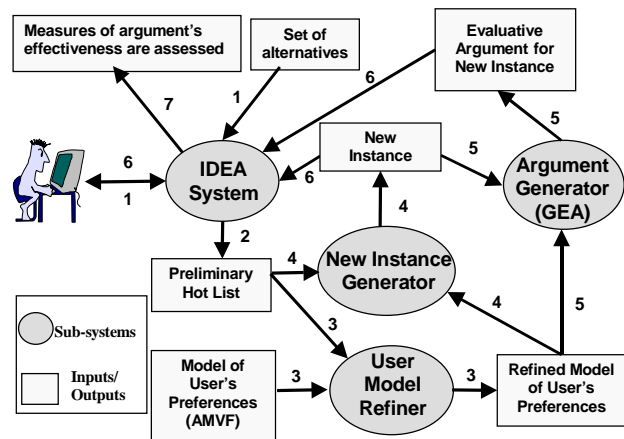


Figure 2 The evaluation framework architecture

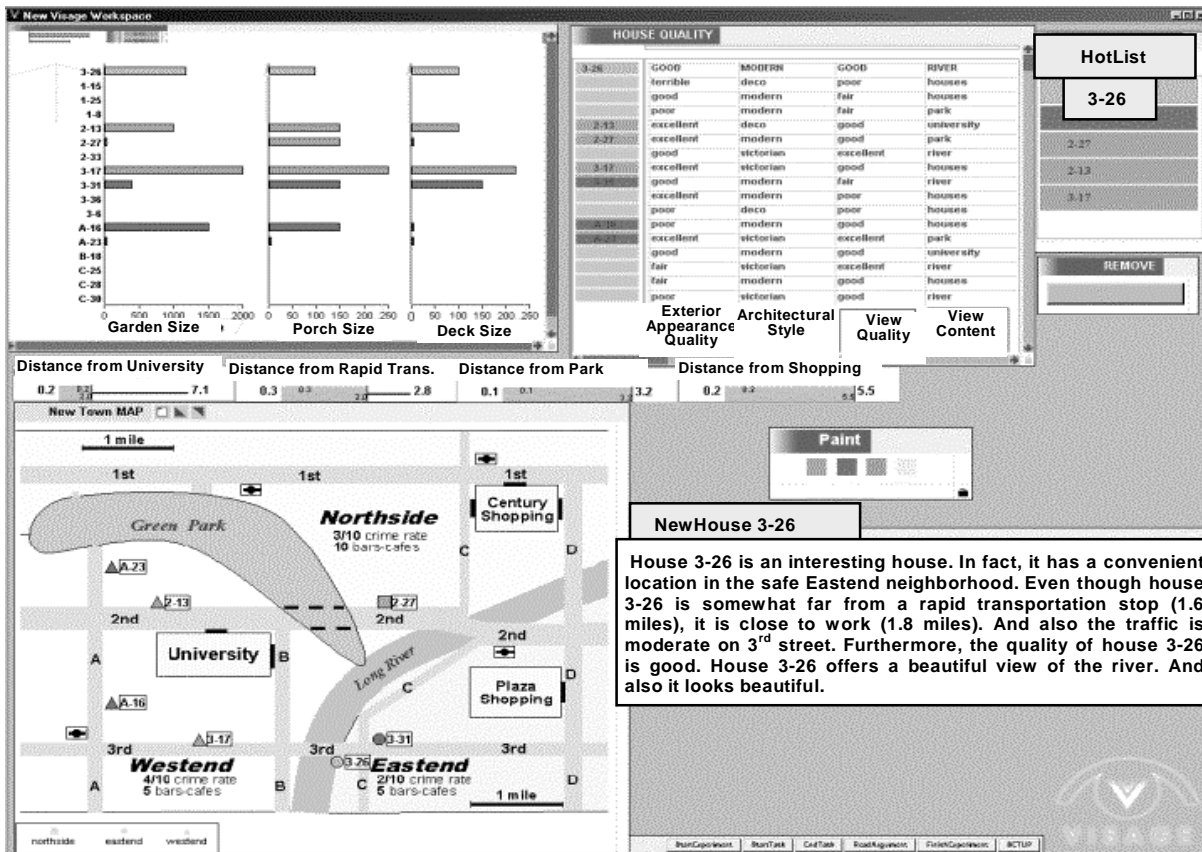


Figure 3 The IDEA environment display at the end of the interaction

- A measure of the user's confidence that she has selected the best for her in the set of alternatives.
- A measure of argument effectiveness derived by explicitly questioning the user at the end of the interaction about the rationale for her decision [Olso and Zanna 1991]. This can provide valuable information on what aspects of the argument were more influential on the user's decision.
- An additional measure of argument effectiveness is derived by explicitly asking the user at the end of the interaction to judge the argument with respect to several dimensions of quality, such as content, organization, writing style and convincingness. However, evaluations based on judgements along these dimensions are clearly weaker than evaluations measuring actual behavioural and attitudinal changes [Olso and Zanna 1991].

To summarize, our evaluation framework supports users in performing a realistic task by interacting with an IDEA system. In the context of this task, an evaluative argument is generated and measurements are collected on its effectiveness. We now discuss an experiment we have performed within the evaluation framework to test to what extent tailoring an evaluative argument to a model of the user preferences increases its effectiveness.

4 The Experiment

Given the goal of our empirical investigation, we have performed a between-subjects experiment with three experimental conditions: (i) *No-Argument* - subjects are simply informed that NewI came on the market. (ii) *Tailored* - subjects are presented with an evaluation of NewI tailored to their preferences. (iii) *Non-Tailored* - subjects are presented with an evaluation of NewI that, instead of being tailored to their preferences, is tailored to the preferences of a default average user, for whom all aspects of a house are equally important (i.e., all weights in the AMVF are the same). A similar default preference model is used for comparative purposes in [Srivastava, Connolly et al. 1995]. In the three conditions, all the information about the NewI is also presented graphically, so that no information is hidden from the subject.

Our hypotheses on the experiment are the following. First, we expect arguments generated for the Tailored condition to be more effective than arguments generated for the Non-Tailored condition. Second, the Tailored condition should be somewhat better than the No-Argument condition, but to a lesser extent, because subjects, in the absence of any argument, may spend more time further exploring the

dataset, thus reaching a more informed and balanced decision. Finally, we do not have strong hypotheses on comparisons of argument effectiveness between the No-Argument and Non-Tailored conditions. The experiment is organized in two phases. In the first phase, the subject fills out a questionnaire on the Web which implements a method from decision theory to acquire an AMVF model of the subject's preferences [Edwards and Barron 1994]. In the second phase, to control for possible confounding variables, including subject's argumentativeness [Infante and Rancer 1982], need for cognition [Cacioppo, Petty et al. 1983], intelligence and self-esteem, the subject is randomly assigned to one of the three conditions. Then, the subject interacts with the evaluation framework and at the end of the interaction measures of the argument effectiveness are collected, as described in Section 3.1.

5 Experiment Results

According to literature on persuasion, the most important measures of argument effectiveness are the ones of behavioral intentions and attitude change [Olso and Zanna 1991]. As explained in Section 3.1, in our framework these measures include (a) whether or not the user adopts NewI, (b) in which position in the Hot List she places it, (c) how much she likes the proposed NewI and the other objects in the Hot List. Measures (a) and (b) are obtained from the record of the user interaction with the system, whereas measures in (c) are obtained from user self-reports.

Figure 4 Sample filled-out self-report on user's satisfaction with houses in the Hot List¹

A closer analysis of the above measures indicates that the measures in (c) are simply a more precise version of measures (a) and (b). In fact, not only do they assess, like (a) and (b), a preference ranking among the new alternative and the other objects in the Hot List, but they also offer two additional critical advantages:

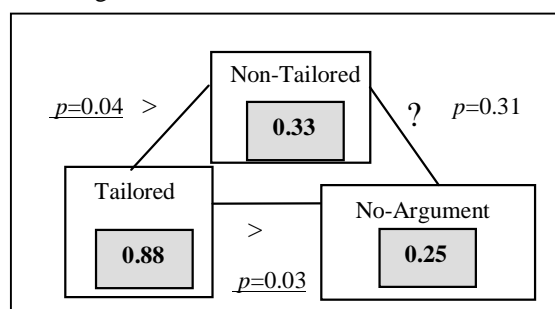


Figure 5 Results for satisfaction z-scores. The average z-scores for the three conditions are shown in the grey boxes

As shown in Figure 5, the satisfaction z-scores obtained in the experiment confirmed our hypotheses. Arguments generated for the Tailored condition were significantly more effective than arguments generated for the Non-Tailored condition ($p=0.04$). The Tailored condition was also

significantly better than the No-Argument condition ($p=0.03$). And this happened despite the fact that subjects in the No-Argument condition spent significantly more time further exploring the dataset after NewI was presented (as indicated in Table 1) ($p=0.05$). Finally, we found no significant difference in argument effectiveness between the No-Argument and Tailored-Verbose conditions.

With respect to the other measures of argument effectiveness mentioned in Section 3.1, we have not found any significant differences among the experimental conditions².

No-Argument	Non-Tailored	Tailored
0:03:56	0:03:37	0:02:44

Table 1 Average time spent by subjects in the three conditions further exploring the dataset after the new house is presented (from Logs of the interaction).

6 Conclusions and Future Work

Argumentation theory indicates that effective arguments should be tailored to a model of the user's preferences. Although previous work on automatically generating evaluative arguments has followed this basic indication, the effect of tailoring on argument effectiveness has never been measured empirically. As an initial attempt to address this issue, we have compared in a formal experiment the effectiveness of arguments that were tailored vs. non-tailored to a model of the user preferences. The experiment results show that tailored arguments are significantly better.

As future work, we plan to perform further experiments. We intend to repeat the same experiment in a different domain to test for external validity. We also envision an experiment in conditions of uncertainty, in which we may compare arguments tailored to an AMVF, with argument tailored to more sophisticated models of user's preferences, that consider interactions among objectives.

References

[Cacioppo, Petty et al. 1983] Cacioppo, J. T., R. E. Petty, et al. Effects of Need for Cognition on Message Evaluation, Recall, and Persuasion. *Journal of Personality and Social Psychology* 45(4): 805-818, 1983.

[Carenini 2000] Carenini, G. Generating and Evaluating Evaluative Arguments. Ph.D. Thesis, Intelligent System Program, University of Pittsburgh, 2000.

[Carenini 2000] Carenini, G. A Task-based Framework to Evaluate Evaluative Arguments. *First International Conference on Natural Language Generation*, Mitzpe Ramon, Israel: 9-16, 2000.

[Carenini and Moore 2000] Carenini, G. and J. Moore. A Strategy for Generating Evaluative Arguments. *First International Conference on Natural Language Generation*, Mitzpe Ramon, Israel: 47-54, 2000.

[Chai, Budzikovaska et al. 2000] Chai, J., M. Budzikovaska, et al. Natural Language Sales Assistant. *38th Annual Meeting of the Association for Computational Linguistics*: 34-35, 2000.

[Clemen 1996] Clemen, R. T. *Making Hard Decisions: an introduction to decision analysis*. Belmont, California, Duxbury Press, 1996.

[Edwards and Barron 1994] Edwards, W. and F. H. Barron. SMARTS and SMARTER: Improved Simple Methods for Multiattribute Utility Measurements. *Organizational Behavior and Human Decision Processes* 60: 306-325, 1994.

[Elhadad, McKeown et al. 1997] Elhadad, M., K. McKeown, et al. Floating constraints in lexical choice. *Computational Linguistics* 23(2): 195-239, 1997.

[Elzer, Chu-Carroll et al. 1994] Elzer, S., J. Chu-Carroll, et al. Recognizing and Utilizing User Preferences in Collaborative Consultation Dialogues. *Proc. of Fourth Int. Conf. of User Modeling*, Hyannis, MA: 19-24, 1994.

[Infante and Rancer 1982] Infante, D. A. and A. S. Rancer. A Conceptualization and Measure of Argumentativeness. *Journal of Personality Assessment* 46: 72-80, 1982.

[Jameson, Schafer et al. 1995] Jameson, A., R. Schafer, et al. Adaptive provision of Evaluation-Oriented Information: Tasks and techniques. *Proceedings of 14th IJCAI*, Montreal, 1995.

[Klein 1994] Klein, D. *Decision Analytic Intelligent Systems: Automated Explanation and Knowledge Acquisition*, Lawrence Erlbaum Associates, 1994.

[Mayberry and Golden 1996] Mayberry, K. J. and R. E. Golden. *For Argument's Sake: A Guide to Writing Effective Arguments*, Harper Collins, College Publisher, 1996.

[Morik 1989] Morik, K. *User Models and Conversational Settings: Modeling the User's Wants*. User Models in Dialog Systems. A. Kobsa and W. Wahlster, Springer-Verlag: 364-385, 1989.

[Olso and Zanna 1991] Olso, J. M. and M. P. Zanna. *Attitudes and beliefs: Attitude change and attitude-behavior consistency*. Social Psychology. R. M. Baron and W. G. Graziano, 1991.

[Solomon 1998] Solomon, M. R. *Consumer Behavior: Buying, Having, and Being*, Prentice Hall, 1998.

[Srivastava, Connolly et al. 1995] Srivastava, J., T. Connolly, et al. Do Ranks Suffice? A Comparison of Alternative Weighting Approaches in Value Elicitation. *Organizational Behavior and Human Decision Process* 63(1): 112-116, 1995.

² The measure of decision rationale is still under analysis.

NATURAL LANGUAGE PROCESSING AND INFORMATION RETRIEVAL

STATISTICAL PROCESSING
OF NATURAL LANGUAGE GRAMMARS

Refining the Structure of a Stochastic Context-Free Grammar

Joseph Bockhorst^{†‡}

joebock@cs.wisc.edu

Mark Craven^{†‡}

craven@biostat.wisc.edu

[†]Department of Computer Sciences

University of Wisconsin

Madison, Wisconsin 53706

[‡]Department of Biostatistics & Medical Informatics

University of Wisconsin

Madison, Wisconsin 53706

Abstract

We present a machine learning algorithm for refining the structure of a stochastic context-free grammar (SCFG). This algorithm consists of a heuristic for identifying structural errors and an operator for fixing them. The heuristic identifies nonterminals in the model SCFG that appear to be performing the function of two or more nonterminals in the target SCFG, and the operator attempts to rectify this problem by introducing a new nonterminal. Structural refinement is important because most common SCFG learning methods set the probability parameters while leaving the structure of the grammar fixed. Thus, any structural errors introduced prior to training will persist. We present experiments that show our approach is able to significantly improve the accuracy of an SCFG designed to model an important class of RNA sequences called *terminators*.

1 Introduction

Stochastic context-free grammars (SCFGs) have long been used in the NLP community (where they are more commonly known as probabilistic context-free grammars) and recently have been applied to biological sequence analysis tasks, in particular RNA analysis. One typical learning problem is to create an SCFG from a set of unparsed training sequences that will accurately recognize previously unseen sequences in the class being modeled. The first step of this two step process is to create the underlying context-free grammar, which we refer to as the *structure* of the model. The second step, the assignment of the probability parameters, is usually performed through an application of an expectation maximization (EM) algorithm called the inside-outside algorithm [Lari & Young, 1990]. One of the limitations of this approach is that if the structure is incomplete or in error, there may be no assignment of probabilities that would result in an accurate model. This is the problem we address in this paper. We introduce a refinement step that can identify and fix a class of structural errors that occur when a nonterminal in the model grammar is performing the function of more than one nonterminal in the target. The particular location refined at each step is determined “diagnostically” by analyzing the interaction between the model and a training set. This method may be contrasted

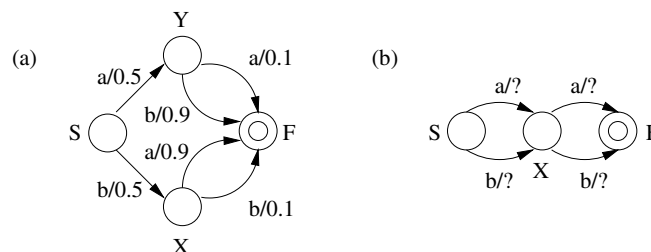


Figure 1: An example where an incorrect structure prevents the learning of an accurate model. (a) A target grammar to be learned. (b) The structure used to model sequences generated by the grammar shown in (a). State S is the begin state and F the end state. Arcs are labeled with the character emitted and the probability of taking that arc. The question marks in (b) represent the probability parameters to be learned. Because state X is overloaded in the learned model, it will be unable to correctly model the probabilities of the second character in the sequence.

to the standard state-space search methodology of applying the best of several competing operations as determined by a heuristic function applied to the successor states. This refinement step can be iterated with the inside-outside algorithm to refine both the structure of the model and the probability parameters. Through experiments with an SCFG designed to model a family of RNA sequences called *terminators* we show how this method may be used to learn more accurate models than inside-outside alone.

To see the impact a structural error can have, consider trying to learn the simple stochastic regular grammar shown in figure 1(a). Imagine that we have many sequences generated by this grammar for training and the structure we choose is shown in figure 1(b). Given enough training data, the maximum likelihood estimate of each state transition probability is 0.5. The probability of each length-two sequence under this model is therefore 0.25 resulting in a rather inaccurate model. The problem with this structure is that state X in the model is overloaded; it is performing the function of both state X and Y in the target. With the approach presented herein, errors such as this can potentially be corrected.

2 Problem Domain

The application of SCFGs we consider here is that of modeling a class of RNA sequences called *terminators*. RNA

sequences are strings from the alphabet $\{a,c,g,u\}$ corresponding to the four different types of bases in the nucleotides of RNA molecules. The bases a and u are *complementary* to one another, as are c and g, because they are able to easily form *base pairs* by becoming chemically bonded¹.

The three dimensional shape that an RNA sequence assumes to a large degree determines its function, and the shape itself is strongly influenced by which particular bases are paired. For example, an element of RNA structure called a *stem* is formed when a number of adjacent bases pair with a complementary set of bases later in the sequence. If the intervening bases are unpaired, the resulting structure is called a *stem-loop*. Figure 2(a) shows an RNA sequence that could assume a stem-loop, and 2(b) shows the stem-loop it forms. The consequence of this is that families of RNA sequences of similar function will share a pattern of dependency between bases in the sequence that are likely to be paired.

Terminators are RNA sequences which signal when to stop the process of *transcription*, one of the key steps in the expression of a gene to form a protein.

3 Stochastic Context Free Grammars

The structure of an SCFG G is a context-free grammar and is defined by a set of nonterminals \mathcal{N} , a set of terminal symbols \mathcal{T} , a start nonterminal $S \in \mathcal{N}$ and a set of rewrite rules, or productions, of the form $N_v \rightarrow \Gamma$, $\Gamma \in \{(\mathcal{N} \setminus S) \cup \mathcal{T}\}^*$. That is, the right hand side of a production consists of any combination of terminal and nonterminal symbols other than the start nonterminal. When we refer to N_v 's productions, we mean the productions with N_v on the left hand side. Nonterminals and terminals are denoted by upper and lower case letters respectively. A grammar can be made into an SCFG by associating with each nonterminal N_v a probability distribution, \mathbf{p}_v , over N_v 's productions.

SCFGs have proven important in RNA analysis tasks because the defining characteristics of RNA families manifest themselves in sequences through long range dependencies between bases. Such dependencies are troublesome to represent with regular grammars but can be naturally represented by context-free grammars. Figure 2(c) shows an SCFG for a specific set of stem-loops and Figure 2(d) shows the parse tree for an example sequence under this grammar.

There are three fundamental computational tasks relating SCFGs and sequences.

1. Compute $Pr(x^i|G)$, the probability of a sequence given a full model.
2. Find the most probable parse tree for a sequence.
3. Given G_s and a set of sequences $\mathcal{X} = \{x^1, x^2, \dots, x^N\}$, set the probabilities to maximize the likelihood $\prod_{i=1}^N Pr(x^i|G)$.

Briefly, the first and second problems are efficiently solvable with dynamic programming algorithms similar to the forward and Viterbi algorithms, but there is no known efficient global solution for the third problem. The most common

¹Base pairs can be formed between non-complementary bases but this is less common.

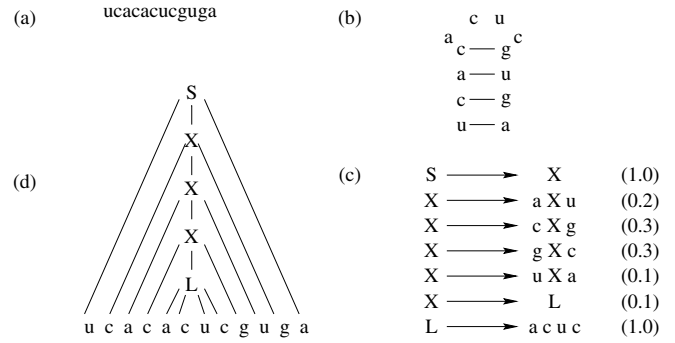


Figure 2: (a) A simple RNA sequence that will form a stem-loop structure. (b) The stem loop structure; paired bases are connected by dashes. (c) An SCFG model of sequences that form stem-loops. Probabilities are listed to the right of their associated production. (d) The parse tree for the sequence in (a) using the grammar in (c).

heuristic is the inside-outside [Lari & Young, 1990] method which, like other EM algorithms, converges to a local maximum.

The inside algorithm, used to solve task 1 above, is a dynamic programming algorithm that fills a three dimensional matrix α . If run on the sequence x^i , the elements of the matrix, $\alpha(j, k, v)$, contain the sum of the probabilities of all parse trees of the subsequence $x_j^i \dots x_k^i$ rooted at N_v . So, if the length of x^i is L and $N_1 = S$, then $\alpha(1, L, 1) = Pr(x^i|G)$.

The partner of the inside algorithm, the outside algorithm calculates a similar dynamic programming matrix β . An element $\beta(j, k, v)$ is the sum of the probabilities of all parse trees of the complete sequence x^i rooted at the start nonterminal, excluding all parse subtrees of $x_j^i \dots x_k^i$ rooted at N_v .

From α and β the expected number of times each nonterminal is used in the derivation of a sequence can be determined. The sum of these counts over all sequences in a training set are used by the inside-outside algorithm to iteratively re-estimate the production probabilities. In the next section we will show how α and β play a role in identifying overloaded nonterminals.

4 Grammar Refinement Approach

When the structure of a hypothesis grammar G^h is different from the structure of the target grammar G^* , its accuracy may suffer. We would like to be able to identify and fix grammars in this situation using only a set of training sequences. In this section, we present a grammar refinement operator and a heuristic to do this. The operator is able to fix certain structural errors in G^h that occur when a nonterminal in the hypothesis is performing the function of more than one nonterminal of G^* . We refer to such a nonterminal in G^h as *overloaded*. The heuristic identifies overloaded nonterminals by locating data dependencies that are not represented by the structure of G^h .

Before we delve into the details of the grammar refinement algorithm, it is helpful to consider the ways in which a hypothesis grammar may differ from the target grammar. Figure 3 gives a taxonomy of possible errors in a hypothesis grammar. The right branch out of the root contains all

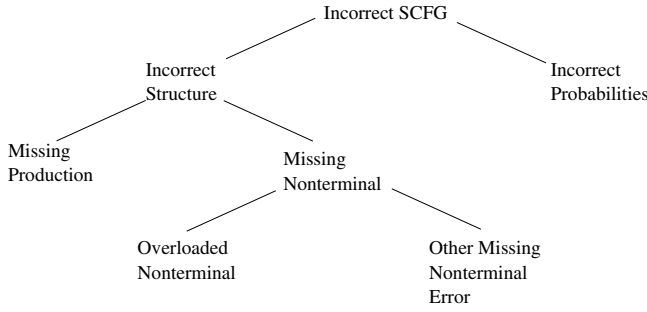


Figure 3: Possible inaccuracies of a hypothesis SCFG with respect to a target grammar \mathcal{G}^*

Table 1: The EXPAND operator. The operator creates a new nonterminal to take the place of nonterminal N on the right hand side of production P

EXPAND(P, N) /* N is on the RHS of production P */

1. Create new nonterminal N'
2. Replace N with N' in P
3. For each production $N \rightarrow \Gamma$, create $N' \rightarrow \Gamma$
4. Initialize probabilities for new productions

SCFGs that have a correct structure; that is, there is some assignment of probabilities that results in a grammar equivalent to \mathcal{G}^* . Most learning algorithms for SCFGs, including inside–outside are aimed at this situation.

The left branch contains two categories of incorrectly structured grammars: those whose structures can be made correct by adding only productions and those that require additional nonterminals as well. Among the kinds of errors that can occur from missing nonterminals, the class we address occurs when a nonterminal in the hypothesis is overloaded because it is trying to perform the function of two or more nonterminals in the target. For example, the nonterminal X in the SCFG of Figure 2(c) would be overloaded if the distribution of the first base pair in the family of sequences being modeled was different from the rest. Next, we present a grammar refinement algorithm to identify and fix overloaded nonterminals.

The grammar refinement problem starts with set a sequences \mathcal{X} which we can think of as generated by an unknown target SCFG \mathcal{G}^* , an initial hypothesis grammar, and a goal of discovering a model ‘close’ to \mathcal{G}^* . We navigate the space of grammar structures through the application of a grammar modification operator, applied at each step to a part of the current grammar chosen by a heuristic.

4.1 Refinement Operator

Our current grammar refinement procedure has a single operator, EXPAND, shown in Table 1, which is applied to what we call a *context*. We define a context to be a production and a nonterminal on the right hand side of that production. We denote a context either by the pair (P, N) where N is a nonterminal on the right hand side of production P or by c_{vr} where r indexes into the productions in which nontermi-

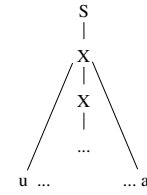


Figure 4: A partial parse tree where the nonterminal X is used in the two different contexts $(S \rightarrow X, X)$ and $(X \rightarrow a X u, X)$.

Table 2: The structure of the SCFG formed following an application of EXPAND($S \rightarrow X, X$) to the grammar in Figure 2(c). The new nonterminal X' replaces X in the production of the context being expanded, $S \rightarrow X$. Productions created by EXPAND are shown in boldface.

$S \rightarrow X'$	
$X \rightarrow a X u$	$X' \rightarrow a X u$
$X \rightarrow c X g$	$X' \rightarrow c X g$
$X \rightarrow g X c$	$X' \rightarrow g X c$
$X \rightarrow a X u$	$X' \rightarrow u X a$
$X \rightarrow L$	$X' \rightarrow L$
$L \rightarrow a c u c$	

nal N_v appears on the right hand side. Every nonterminal node in a parse tree defines a context. For example, the nonterminal X in the parse tree fragment of Figure 4 is in the context $(S \rightarrow X, X)$ the first time it appears and $(X \rightarrow a X u, X)$ the second. When applied to (P, N) , EXPAND creates a new nonterminal N' , replaces N with N' in P and creates a production with N' on the left hand side from each of N ’s productions. Table 2 shows the structure that is formed after applying EXPAND($S \rightarrow X, X$) to the SCFG of Figure 2(c). We explain how to set the probabilities of the new productions below.

4.2 Refinement Heuristics

The heuristic we use to guide the search expands a single context at each step. It chooses to expand the nonterminal that is used most differently in that context compared to its expected usage based on its probability distribution. Let \mathbf{n}_{vr} be the vector of expected usage counts of the productions of N_v in context c_{vr} (i.e., on the right hand side of the production indicated by r) for the training sequences and let n_{vrl} be an element of this vector that refers to the expected number of times that the l^{th} production with N_v on the left hand side is used in context c_{vr} . EXPAND(c_{vr}) sets the probability distribution of the nonterminal it creates to the one defined by \mathbf{n}_{vr} . Given a measurement of difference between the data distribution n_{vr} and N_v ’s probability distribution \mathbf{p}_v we define our heuristic to choose to expand the context that maximizes this difference. We have investigated two difference measures, one based on the Kullback–Leibler (KL) divergence and the other based on the χ^2 statistic.

The KL divergence, also called relative entropy, between two probability distributions \mathbf{p} and \mathbf{q} with the same discrete

Table 3: Example of values used in computing the heuristics for the context $(S \rightarrow X, X)$ from the SCFG in Figure 2(c). The columns indicate: (1) the productions of X , (2) the observed number of times it is used in the context, (3) the observed probability distribution in the context, (4) the expected usage counts if the context’s distribution were the same as the distribution over all contexts, (5) the distribution over all contexts.

Production	Context $(S \rightarrow X, X)$			All
	“Observed”		Expected	
	Counts	Pr	Counts	Pr
	\mathbf{n}_{vr}	\mathbf{p}_{vr}	\mathbf{e}_{vr}	\mathbf{p}_v
$X \rightarrow a X u$	5.0	0.10	10.0	0.2
$X \rightarrow c X g$	26.5	0.53	15.0	0.3
$X \rightarrow g X c$	10.0	0.20	15.0	0.3
$X \rightarrow u X a$	3.5	0.07	5.0	0.1
$X \rightarrow L$	5.0	0.10	5.0	0.1
Sum	50.0	1.00	50.0	1.0

domain is defined as

$$KL(\mathbf{p}, \mathbf{q}) = \sum_i p_i \log_2 \frac{p_i}{q_i}.$$

Consider the example in Table 3. Column (3) in this table shows \mathbf{p}_{vr} for the productions of X in the context $(S \rightarrow X, X)$. We can use KL divergence to compare this distribution to \mathbf{p}_v , shown in column (5), which is the probability distribution for the productions of X over all of X ’s contexts. Using this measure, we select the nonterminal N_v in context r that maximizes

$$KL(\mathbf{p}_{vr}, \mathbf{p}_v) \sum_l n_{vrl}$$

where \mathbf{p}_{vr} is the probability distribution defined by \mathbf{n}_{vr} , and the sum calculates the expected total number of times N_v is in c_{vr} . By one interpretation, this measure selects the context that would waste the most bits in transmitting a message for each production applied in that context if the encoding used is the optimal encoding defined by \mathbf{p}_v instead of \mathbf{p}_{vr} .

The second heuristic we consider, the χ^2 statistic, is commonly used to determine if the hypothesis that a data sample was drawn according to some distribution can be rejected. Using the vector of expected usage counts of nonterminal N_v in context c_{vr} , \mathbf{n}_{vr} , as the “observed” counts and $\mathbf{e}_{vr} = \mathbf{p}_v * (\sum_l n_{vrl})$ for the expected counts,

$$\chi_{vr}^2 = \sum_l ((e_{vrl} - n_{vrl})^2 / e_{vrl}).$$

Our method selects for expansion the context c_{vr} which maximizes χ_{vr}^2 . Returning to the example in Table 3, we can use χ^2 to assess the significance of differences between column (2), the observed counts for the productions applied in the given context, and column (4) the expected observations given the probabilities in column (5).

To apply either of these heuristics, we must first calculate the counts n_{vrl} . These can be calculated from the inside and outside matrices α and β in a similar way as the usage counts of productions and nonterminals are calculated in the

Table 4: The structure of our terminator grammar. Nonterminals are capitalized and the terminals are a, c, g and u. The notation $X \rightarrow Y \mid Z$ is shorthand for the two productions $X \rightarrow Y$ and $X \rightarrow Z$. Productions containing t_l and t_r are shorthand for the family of 16 productions where t_l and t_r can be any of the four terminals. Productions containing t_l^* and t_r^* have a similar interpretation except that t_l^* and t_r^* can also be null allowing for unpaired bases in the interior of the stem.

START	\rightarrow	PREFIX STEM_BOT1 SUFFIX
PREFIX	\rightarrow	B B B B B B B
STEM_BOT1	\rightarrow	t_l STEM_BOT2 t_r
STEM_BOT2	\rightarrow	t_l^* STEM_MID t_r^* t_l^* STEM_TOP2 t_r^*
STEM_MID	\rightarrow	t_l^* STEM_MID t_r^* t_l^* STEM_TOP2 t_r^*
STEM_TOP2	\rightarrow	t_l^* STEM_TOP1 t_r^*
STEM_TOP1	\rightarrow	t_l LOOP t_r
LOOP	\rightarrow	B B LOOP_MID
LOOP_MID	\rightarrow	B LOOP_MID B B
SUFFIX	\rightarrow	B B B B B B B
B	\rightarrow	a c g u

inside–outside algorithm. For example, the expected number of times that the production $N_w \rightarrow c N_y g$ is used in the context $(N_v \rightarrow a N_w u, N_w)$ in the derivation of the sequence x^i is

$$Pr(N_v \rightarrow a N_w u) Pr(N_w \rightarrow c N_y g) \sum_j \sum_k I \gamma_{jk}$$

where

$$\gamma_{jk} = \beta(j, k, v) \alpha(j + 2, k - 2, y)$$

and I is an indicator variable that is 1 if $x_j^i = a, x_{j+1}^i = c, x_{k-1}^i = g$ and $x_k^i = u$, and 0 otherwise. Also, the counts are smoothed by adding 1 to each n_{vrl} .

5 Experiments

We have constructed an SCFG model of terminator sequences by incorporating known features of terminators identified in the terminator literature. The structure of the terminator grammar we start with, shown in Table 4, models the eight bases before and eight bases after the stem–loop and the stem–loop itself. The recursive nature of STEM_MID and LOOP_MID’s productions enables variable length stems and loops to be represented. An unpaired base in the interior of the stem, a *bulge*, can also be represented.

Positive examples consist of the sequences of 142 known or proposed terminators of *E. coli*. Each of these sequences is 50 base pairs long with the 30th base aligned to the assumed termination point. We have 125 negative sequences, also of length 50, which were collected from the regions following genes that are presumed to not contain terminators.

We performed a five fold cross validation experiment using the KL and χ^2 heuristic as well as a control where the context to EXPAND at each step is randomly chosen. The following steps were repeated.

1. Extract \mathcal{T} , the most probable subsequence of each positive sequence in the training set given the current model.

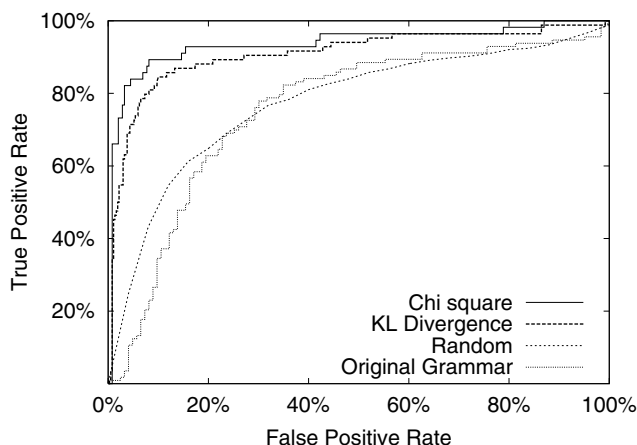


Figure 5: ROC curves of original grammar and the refined grammars after 25 additional nonterminals have been added using the χ^2 , KL divergence and random heuristics. Note that the order of the models in the key reflects the order of the curves.

2. Fit the probability parameters by running inside–outside to convergence on \mathcal{T} .
3. EXPAND the context chosen by the heuristic.

The first step is needed because the positive sequences contain more bases than what is being modeled by the grammar. The probabilities in the stem were initialized with a preference for complementary base pairs; elsewhere, a uniform distribution was used.

For each test sequence, we determine the probability of its most probable subsequence as given by each model. To compare various methods, we rank our test-set predictions by these probability values and then construct ROC curves. We get a single ROC curve for each approach by pooling predictions across test-sets.

Figure 5 shows the ROC curves that result from the original grammar, and from running our grammar refinement algorithm for 25 iterations using the χ^2 , KL, and random heuristics. The random curve is the average over 35 random runs. The curves in this figure show that our heuristics provide better predictive accuracy than both the original grammar and our control of randomly selecting the context to expand at each step. These results support our hypothesis that directed changes to the structure of the grammar can result in more accurate learned models.

To consider how the predictive accuracy of the grammars changes as a function of the number of refinement iterations, we construct ROC curves after each iteration and then calculate the area under each curve. Figure 6 plots the area under these curves versus the number of nonterminals added for the three heuristics considered in Figure 5. The y -axis for this figure starts at 0.5 which is the expected area under the curve for a model that guessed randomly (such a model would result in an ROC “line” defined by: TP rate = FP rate). The results in this figure show that steady improvement is seen through about the 15th additional nonterminal using the KL heuristic and through about the 20th using χ^2 . Our approach does not appear to overfit the training data, at least through

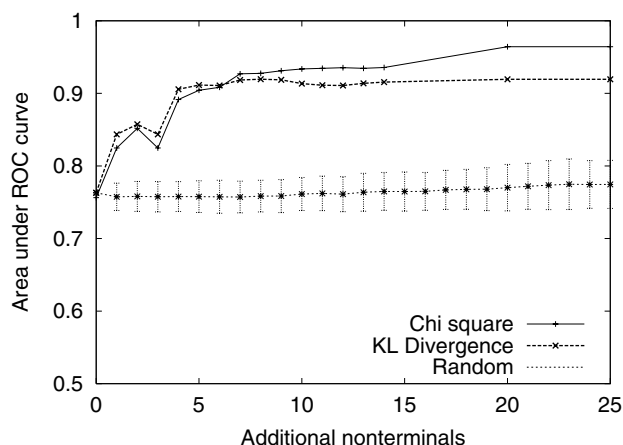


Figure 6: Plot of the area under the ROC curves for three heuristics: χ^2 , KL Divergence, and random. The error bars on the random plot denote the 95% confidence interval. Note that the y -axis starts at 0.5 which corresponds to the expected area under the curve for random predictions.

the 25 iterations for which we have run it.

One possible reason that the χ^2 heuristic results in more accurate models than the KL heuristic is the following. We know that our calculations of KL Divergence are biased because they are calculated with finite samples [Herzel & Grosse, 1997], but that this bias is not uniform across the nonterminals and contexts we are comparing because the sample sizes differ. Therefore, one avenue for future research is to investigate measures that correct for the finite-sample bias of our KL Divergence calculation.

We have examined the sequence of contexts expanded for both the χ^2 and the KL Divergence heuristics, and noticed that several of the early refinements adjust a part of the grammar that describes the first few positions following the stem. There is known preference for t bases in this part of terminators, but our initial grammar did not encode it. This result suggests that our method can compensate for a known inadequacy in the initial grammar. However, we note that the algorithm makes additional modifications to other parts of the initial grammar that represent our best effort at encoding relevant domain knowledge.

One of the limitations of this algorithm as presented is the one-way nature of the search. That is, there is no way to make the grammar more general. If the initial grammar encodes the most general knowledge of the domain expert, this limitation may not be problematic initially. However, as EXPAND creates more nonterminals, it is likely that the grammar would become overly specific in places. The most obvious way of addressing this problem is by introducing a generalization operator. For example, a MERGE operator could be used to combine two nonterminals with the same RHS domain if their probability distributions are sufficiently similar. Note however, as mentioned by Stolcke (1994), that this operator will not introduce any new embedding structure into the grammar. This may not be a problem if such structure is present in the initial grammar.

6 Related Work

The approach we present here is related to the grammar induction algorithms of Stolcke (1994) and Chen (1996). These works both address the induction of SCFGs from text corpora for use as language models. Both methods incorporate a prior probability distribution over model structures and perform a search through posterior model probability space where Stolcke proceeds specific to general and Chen general to specific. One of the key differences between these algorithms and ours is that each of these addresses *tabula rasa* grammar induction while ours begins with a grammar structure created from prior knowledge. Another key difference is the way in which steps in the search space are selected. The methods of both Stolcke and Chen evaluate a candidate operator application by doing one-step lookahead. Since it can be expensive to calculate the posterior probability of the data given the changed model, heuristics are used to estimate this value. The operator application that results in the greatest estimated posterior probability is accepted. Our approach, on the other hand, is more “diagnostic.” Instead of doing one-step lookahead, our heuristics try to directly identify places in which the grammar structure is inadequate. This approach may somewhat insulate our method from the overfitting pitfalls of likelihood driven searches. A third difference is that Stolcke and Chen use a richer set of operators than we do. As suggested in the previous section, however, we believe that our diagnostic approach can be generalized to work with other operators.

A different view of our approach is as an instance of a theory refinement algorithm [Pazzani & Kibler, 1992; Ourston & Mooney, 1994; Towell & Shavlik, 1994]. In theory refinement, the goal is to improve the accuracy of an incomplete or incorrect domain theory, from a set of labeled training examples. The primary difference between our work and previous work in theory refinement is the representation used by our learned models. Whereas previous theory-refinement methods have focused on logic-based and neural network representations, our learned models are represented using stochastic context free grammars.

The task we address is also similar to the problem of learning the structure of Bayesian networks [Heckerman, Geiger, & Chickering, 1995; Chickering, 1996], where the statistical properties of a training set are used to guide the modifications of the network structure. Again, the principal difference between our approach and this body of work is the difference in the representation language.

There has also been related work in learning SCFGs for RNA modeling tasks [Eddy & Durbin, 1994; Sakakibara *et al.*, 1994]. These iterative methods both update grammar structure using information gathered from the most probable parse of each of the training sequences at each step. Although we intend to compare our refinement algorithm to both of these approaches in future work, we consider our work to be complementary to these methods because it could be run on every one of their iterations.

7 Conclusion

We have considered the problem of refining the structure of a deficient SCFG using a set of training sequences. We intro-

duced a grammar refinement operator, EXPAND, for repairing a type of SCFG structural error resulting from an overloaded nonterminal as well as a pair of heuristics, based on KL divergence and χ^2 , for locating them. Preliminary results indicate that our method is able to improve the accuracy of an SCFG designed to model terminators by correcting known structural errors as well making modifications to parts of the grammar with no known deficiencies.

Acknowledgments

This research was supported in part by NSF CAREER award IIS-0093016 and NIH Grant 1R01 LM07050-01. Thanks to Soumya Ray for helpful comments on an earlier version of this paper.

References

- [Chen, 1996] Chen, S. 1996. *Building Probabilistic Models for Natural Language*. Ph.D. Dissertation, Harvard University.
- [Chickering, 1996] Chickering, D. M. 1996. Learning equivalence classes of Bayesian-network structure. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann.
- [Eddy & Durbin, 1994] Eddy, S. R., and Durbin, R. 1994. RNA sequence analysis using covariance models. *Nucleic Acids Research* 22:2079–2088.
- [Heckerman, Geiger, & Chickering, 1995] Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks. *Machine Learning* 20:197–243.
- [Herzel & Grosse, 1997] Herzel, H., and Grosse, I. 1997. Correlations in DNA sequences: The role of protein coding segments. *Physical Review E* 55(1).
- [Lari & Young, 1990] Lari, K., and Young, S. J. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language* 4:35–56.
- [Ourston & Mooney, 1994] Ourston, D., and Mooney, R. 1994. Theory refinement combining analytical and empirical methods. *Artificial Intelligence* 66(2):273–309.
- [Pazzani & Kibler, 1992] Pazzani, M., and Kibler, D. 1992. The utility of knowledge in inductive learning. *Machine Learning* 9(1):57–94.
- [Sakakibara *et al.*, 1994] Sakakibara, Y.; Brown, M.; Hughey, R.; Mian, I. S.; Sjölander, K.; Underwood, R. C.; and Haussler, D. 1994. Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research* 22:5112–5120.
- [Stolcke, 1994] Stolcke, A. 1994. *Bayesian Learning of Probabilistic Language Models*. Ph.D. Dissertation, University of California, Berkeley.
- [Towell & Shavlik, 1994] Towell, G., and Shavlik, J. 1994. Knowledge-based artificial neural networks. *Artificial Intelligence* 70(1,2):119–165.

Automatically Extracting and Comparing Lexicalized Grammars for Different Languages

Fei Xia, Chung-hye Han, Martha Palmer, and Aravind Joshi

University of Pennsylvania
Philadelphia PA 19104, USA

{fxia, chunghye, mpalmer, joshi}@linc.cis.upenn.edu

Abstract

In this paper, we present a quantitative comparison between the syntactic structures of three languages: English, Chinese and Korean. This is made possible by first extracting Lexicalized Tree Adjoining Grammars from annotated corpora for each language and then performing the comparison on the extracted grammars. We found that the majority of the core grammar structures for these three languages are easily inter-mappable.

1 Introduction

The comparison of the grammars extracted from annotated corpora (i.e., Treebanks) is important on both theoretical and engineering grounds. Theoretically, it allows us to do a quantitative testing of the Universal Grammar Hypothesis. One of the major concerns in modern linguistics is the establishment of an explanatory basis for the similarities and variations among languages. The working assumption is that languages of the world share a set of universal linguistic principles and the apparent structural differences attested among languages can be explained as variation in the way the universal principles are instantiated. Comparison of the extracted syntactic trees allows us to quantitatively evaluate how similar the syntactic structures of different languages are. From an engineering perspective, the extracted grammars and the links between the syntactic structures in the grammars are valuable resources for NLP applications, such as parsing, computational lexicon development, and machine translation (MT), to name a few.

In this paper we first briefly discuss some linguistic characteristics of English, Chinese, and Korean, and introduce the Treebanks for the three languages. We then describe a tool that extracts Lexicalized Tree Adjoining Grammars (LTAGs) from the Treebanks and the results of its application. Next, we describe our methodology for automatic comparison of the extracted Treebank grammars, which consists primarily of matching syntactic structures (namely, templates, sub-templates and context-free rules) in each pair of Treebank grammars. The ability to perform this type of comparison for different languages enables us to distinguish language-independent features from language-dependent ones. Therefore, our grammar extraction tool is not only an engineering

tool of great value in improving the efficiency and accuracy of grammar development, but it is also very useful for investigating theoretical linguistics.

2 Our Annotated Corpora

In this section, we briefly discuss some linguistic characteristics of English, Chinese, and Korean, and introduce the Treebanks for these languages.

2.1 Differences between the Three Languages

These three languages belong to different language families: English is Germanic, Chinese is Sino-Tibetan, and Korean is Altaic [Comrie, 1987]. There are several major differences between these languages. First, both English and Chinese have predominantly subject-verb-object (SVO) word order, whereas Korean has underlying SOV order. Second, the word order in Korean is freer than in English and Chinese in the sense that argument NPs are freely permutable (subject to certain discourse constraints). Third, Korean and Chinese freely allow subject and object deletion, but English does not. Fourth, Korean has richer inflectional morphology than English, whereas Chinese has little, if any, inflectional morphology.

2.2 Treebank Description

The Treebanks that we used in this paper are the English Penn Treebank II [Marcus *et al.*, 1993], the Chinese Penn Treebank [Xia *et al.*, 2000b], and the Korean Penn Treebank [Han *et al.*, 2001]. The main parameters of these Treebanks are summarized in Table 1.¹ The tagsets include four types of tags: Part-Of-Speech (POS) tags for head-level annotation, syntactic tags for phrase-level annotation, function tags for grammatical function annotation, and empty category tags for dropped arguments, traces, and so on.

We chose these Treebanks because they all use phrase structure annotation and their annotation schemata are similar, which facilitates the comparison between the extracted Treebank grammars. Figure 1 shows an annotated sentence from the English Penn Treebank.

¹The reason why the average sentence length for Korean is much shorter than those for English and Chinese is that the Korean Treebank includes dialogues that contain many one-word replies, whereas English and Chinese corpora consist of newspaper articles.

Language	corpus size (words)	ave sentence length	tagset size
English	1,174K	23.85 words	94
Chinese	100K	23.81 words	92
Korean	54K	10.71 words	61

Table 1: Sizes of the Treebanks and their tagsets

((S (PP-LOC (IN at)
 (NP (NNP FNX))
 (NP-SBJ-1 (NNS underwriters))
 (ADVP (RB still))
 (VP (VBP draft)
 (NP (NNS policies))
 (S-MNR
 (NP-SBJ (-NONE- *-1))
 (VP (VBG using)
 (NP
 (NP (NN fountain) (NNS pens))
 (CC and)
 (NP (VBG blotting) (NN papers)))))))))

Figure 1: An example from the English Penn Treebank

3 Extracting Grammars

In this section, we give a brief introduction to the LTAG formalism and to a system named LexTract, which was built to extract LTAGs from the Treebanks [Xia, 1999; Xia *et al.*, 2000a].

3.1 The Grammar Formalism

LTAGs are based on the Tree Adjoining Grammar formalism developed by Joshi and his colleagues [Joshi *et al.*, 1975; Joshi and Schabes, 1997]. The primitive elements of an LTAG are elementary trees (*etrees*). Each *etree* is associated with a lexical item (called the *anchor* of the tree) on its frontier. LTAGs possess many desirable properties, such as the Extended Domain of Locality, which allows the encapsulation of all arguments of the anchor associated with an *etree*. There are two types of *etrees*: initial trees and auxiliary trees. An auxiliary tree represents a recursive structure and has a unique leaf node, called the *foot* node, which has the same syntactic category as the root node. Leaf nodes other than anchor nodes and foot nodes are *substitution* nodes. *Etrees* are combined by two operations: substitution and adjunction. The resulting structure of the combined *etrees* is called a *derived tree*. The history of the combination process is expressed as a *derivation tree*. Figure 2 shows the *etrees*, the derived tree, and the derivation tree for the sentence *underwriters still draft policies*. Foot and substitution nodes are marked by * and ↓, respectively. The dashed and solid lines in the derivation tree are for adjunction and substitution operations, respectively.

3.2 The Target Grammars

Without further constraints, the *etrees* in the target grammar (i.e., the grammar to be extracted by LexTract) could be of various shapes. LexTract recognizes three types of relations between the anchor of an *etree* and other nodes in the *etree*; namely, predicate-argument, modification, and coordination relations. It imposes the constraint that all the *etrees* to be

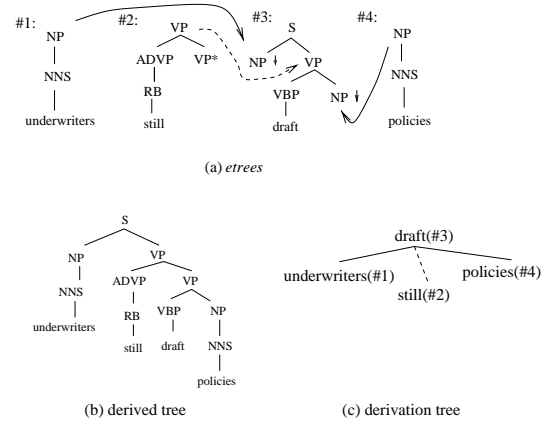


Figure 2: *Etrees*, derived tree, and derivation tree for *underwriters still draft policies*

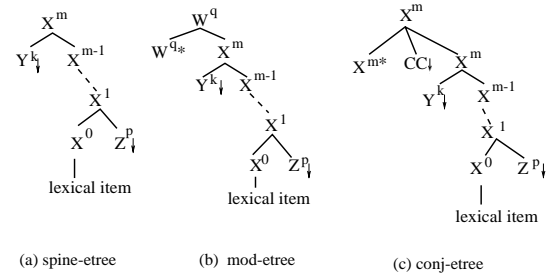


Figure 3: Three types of elementary trees in the target grammar

extracted should fall into exactly one of the three patterns (as in Figure 3):²

Spine-etrees for predicate-argument relations: X^0 is the head of X^m and the anchor of the *etree*. The *etree* is formed by the spine $X^m \rightarrow X^{m-1} \rightarrow \dots \rightarrow X^0$ and the arguments of X^0 .

Mod-etrees for modification relations: The root of the *etree* has two children, one is a foot node with the same label W^q as the root node, and the other node X^m is a modifier of the foot node. X^m is further expanded into a spine-etree whose head X^0 is the anchor of the whole mod-etree.

Conj-etrees for coordination relations: In a conj-etree, the children of the root are two conjoined constituents and a node for a coordinating conjunction. One conjoined constituent is marked as the foot node, and the other is expanded into a spine-etree whose head is the anchor of the whole tree.

Spine-etrees by themselves are initial trees, whereas mod-etrees and conj-etrees are auxiliary trees.

3.3 The LexTract Algorithm

The core of LexTract is an extraction algorithm that takes a Treebank sentence such as the one in Figure 1 and Treebank-specific information provided by the user of LexTract, and

²The precedence relation between the children of the nodes in these three patterns is unspecified, and may vary from language to language.

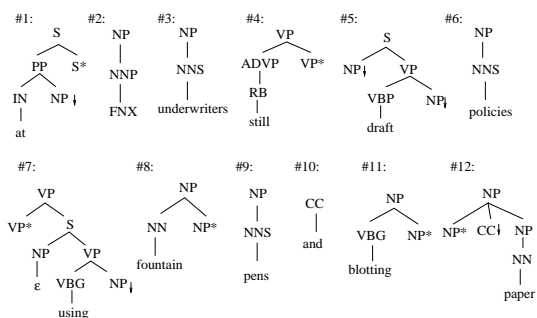


Figure 4: The extracted *etrees* from the phrase structure in Figure 1

produces a set of *etrees* as in Figure 4 and a derivation tree. LexTract’s extraction algorithm has been described in [Xia, 1999] and is completely language-independent. It has been successfully applied to the development of language processing tools such as SuperTaggers [Xia *et al.*, 2000a] and statistical LTAG parsers [Sarkar, 2001].

3.4 Extracted Grammars

The results of running LexTract on English, Chinese, and Korean Treebanks are shown in Table 2. *Templates* are *etrees* with the lexical items removed. For instance, #3, #6, and #9 in Figure 4 are three distinct *etrees*, but they share the same *template*. LexTract is designed to extract LTAGs, but simply reading context-free rules off the templates in an extracted LTAG will yield a context-free grammar. The last column in the table shows the numbers of the non-lexicalized context-free rules.

In each Treebank, a small subset of template types, which occur very frequently in the Treebank and can be seen as members of the core of the Treebank grammar, covers the majority of template tokens in the Treebank. For instance, the top 100 (500, 1000 and 1500, respectively) template types in the English Penn Treebank cover 87.1% (96.6%, 98.4% and 99.0%, respectively) of the tokens, whereas about half (3440) of the template types occur once, accounting for only 0.32% of the template tokens in total.

	template types	<i>etree</i> types	word types	context-free rules
English	6926	131,397	49,206	1524
Chinese	1140	21,125	10,772	515
Korean	632	13,941	10,035	152

Table 2: Grammars extracted from three Treebanks

4 Comparing Treebank Grammars for Different Languages

In this section, we describe our methodology for comparing Treebank grammars and the experimental results.

4.1 Methodology

To compare Treebank grammars, we need to ensure that the Treebank grammars are based on the same tagset. To achieve

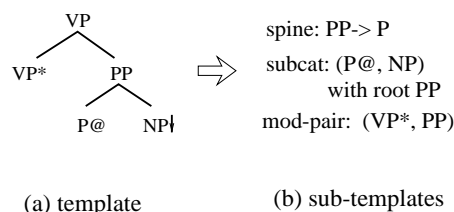


Figure 5: The decomposition of an *etree* template (In sub-templates, @ marks the anchor in subcategorization frame, * marks the modifier in a modifier-modifiee pair.)

that, we first create a new tagset that includes all the tags from the three Treebanks. Then we merge several tags in this new tagset into a single tag.³ Next, we replace the tags in the original Treebanks with the tags in the new tagset, and then re-run LexTract to build Treebank grammars from those Treebanks.

Now that the Treebank grammars are based on the same tagset, we can compare them according to the templates, context-free rules, and sub-templates that appear in more than one Treebank — that is, given a pair of Treebank grammars, we first calculate how many templates occur in both grammars;⁴ Second, we read context-free rules off the templates and compare these context-free rules; Third, we decompose each template into a list of *sub-templates* (e.g., spines and subcategorization frames) and compare these sub-templates. A template is decomposed as follows: A spine-*etree* template is decomposed into a spine and a subcategorization frame; a mod-*etree* template is decomposed into a spine, a subcategorization frame, and a modifier-modifiee pair; a conj-*etree* template is decomposed into a spine, a subcategorization frame, and a coordination tuple. Figure 5 shows the decomposition of a mod-*etree* template.

4.2 Initial Results

After the tags in original Treebanks have been replaced with the tags in the new tagset, the numbers of templates in the new Treebank grammars decrease by about 50%, as shown in the second column of Table 3 (cf. the second column in Table 2). Table 3 also lists the numbers of context-free rules and sub-templates (e.g., spines and subcategorization frames) in each grammar.

The third column of Table 4 lists the numbers of template types shared by each pair of Treebank grammars and the percentage of the template tokens in each Treebank that are covered by these common template types. For example, there

³This step is necessary because certain distinctions among some tags in one language do not exist in another language. For example, the English Treebank has distinct tags for past tense verbs, past participals, gerunds, and so on; however, no such distinction is morphologically marked in Chinese and, therefore, the Chinese Treebank uses the same tag for verbs regardless of the tense and aspect. To make the conversion straightforward for verbs, we use a single tag for verbs in the new tagset.

⁴Ideally, to get more accurate comparison results, we would like to compare *etrees*, rather than templates (which are non-lexicalized); however, comparing *etrees* requires bilingual parallel corpora, which we are currently building.

	<i>templates</i>	context-free rules	<i>subtemplates</i>				total
			spines	subcat frames	mod-pairs	conj-tuples	
Eng	3139	754	500	541	332	53	1426
Ch	547	290	108	180	152	18	458
Kor	256	102	43	65	54	5	167

Table 3: Treebank grammars with the new tagset

		templates	context-free rules	sub-templates
(Eng, Ch)	type (#)	237	154	246
	token (%)	80.1/81.5	88.0/85.2	91.4/85.2
(Eng, Kor)	type (#)	54	61	96
	token (%)	47.6/85.6	53.4/92.2	58.9/98.4
(Ch, Kor)	type (#)	43	44	69
	token (%)	55.9/81.0	63.2/89.3	65.7/96.0

Table 4: Comparisons of templates, context-free rules, and sub-templates in three Treebank grammars

are 237 template types that appear in both English and Chinese Treebank grammars. These 237 template types account for 80.1% of the template tokens in the English Treebank, and 81.5% of the template tokens in the Chinese Treebank. The table shows that, although the numbers of matched templates are not very high, most of these templates have high frequency and therefore account for the majority of the template tokens in the Treebanks. For instance, in the (Eng, Ch) pair, the 237 template types that appear in both grammars is only 7.5% of all the English template types, but they cover 80.1% of the template tokens in the English Treebank.

If we compare sub-templates, rather than templates, the percentages of matched sub-template tokens (as shown in the last column in Table 4) are higher than the percentages of matched template tokens. This is because two distinct templates may have common sub-templates. Similarly, the percentages of matched context-free rules (see the fourth column in Table 4) are higher than the percentages of matched template tokens.

4.3 Results Using Thresholds

The comparison results shown in Table 4 used every template in the Treebank grammars regardless of the frequency of the template in the corresponding Treebank. One potential problem with this approach is that some annotation errors in the Treebanks could have a substantial effect on the comparison results. One such scenario is as follows: To compare languages A and B, we use Treebanks T_A for language A and Treebank T_B for language B. Let G_A and G_B be the grammars extracted from T_A and T_B , respectively, and let t be a template that appears in both grammars. Now suppose that t is a linguistically valid template for language A and it accounts for 10% of the template tokens in T_A , but t is not a valid template for language B and it appears once in Treebank B only due to annotation errors. In this scenario, if G_B excluding template t covers 50% of the template tokens in Treebank A, then G_B including t covers 60% of the template tokens in Treebank A. In other words, the single error in Treebank B, which results in template t being included in G_B , changes the comparison results dramatically.

Because most templates that are due to annotation errors occur very infrequently in the Treebanks, we used a threshold to discard from the Treebanks and Treebank grammars all the templates with low frequency in order to reduce the effect of Treebank annotation errors on the comparison results, Table 5 shows the numbers of templates in the Treebank grammars when the threshold is set to various values. For example, the last column lists the numbers of templates that occur at least 40 times in the Treebanks.

Table 6 shows the numbers of matched templates and the percentages of matched template tokens when the low frequency templates are removed from the Treebanks and Treebank grammars. As the value of the threshold increases, for each language pair the number of matched templates decreases. The percentage of matched template tokens might decrease a fair amount at the beginning, but it levels off after the threshold reaches a certain value. This tendency is further illustrated in Figure 6. In this figure, the X-axis is the threshold value, which ranges from 1 to 40; the Y-axis is the percentage of matched template tokens in each Treebank when the templates with low frequency are discarded. The curve on the top is the percentage of template tokens in the Chinese Treebank that are covered by the English grammar, and the curve on the bottom is the percentage of template tokens in the English Treebank that are covered by the Chinese grammar. Both curves become almost flat once the threshold value reaches 6 or larger. This result implies that most templates due to annotation errors occur less than six times in the Treebanks.

To summarize, in order to get a better estimate of the percentage of matched template tokens, we should disregard the low frequency templates in the Treebanks. We have shown that this strategy reduces the effect of annotation errors on the comparison results (see Table 6 and Figure 6). This strategy also makes the difference between the sizes of our three Treebanks less important because once Treebanks reach a certain size, the new templates extracted from additional data tend to have very low frequency in the whole Treebank.

	1	2	3	4	5	10	20	30	40
English	3139	1804	1409	1209	1065	762	524	444	386
Chinese	547	341	272	226	210	155	122	110	100
Korean	256	181	146	132	122	94	67	57	53

Table 5: The numbers of templates in the Treebank grammars with the threshold set to various values

	threshold	1	2	3	4	5	10	20	30	40
(Eng, Ch)	type (#)	237	165	128	111	100	73	54	47	42
	token (%)	80.1/81.5	76.5/80.8	64.3/80.6	64.1/80.6	63.8/78.5	58.1/77.9	57.3/76.9	57.3/76.0	56.4/76.0
(Eng, Kor)	type (#)	54	47	37	35	32	29	23	21	19
	token (%)	47.6/85.6	47.5/81.0	47.2/81.0	47.3/80.7	47.3/80.7	47.3/80.4	47.5/79.3	47.5/79.4	47.6/79.3
(Ch, Kor)	type (#)	43	36	34	29	27	22	18	18	17
	token (%)	55.9/81.0	56.0/81.0	56.0/77.0	56.1/76.0	55.8/76.1	56.0/74.3	56.1/74.5	56.3/74.9	56.5/74.9

Table 6: Matched templates in the Treebank grammars with various threshold values

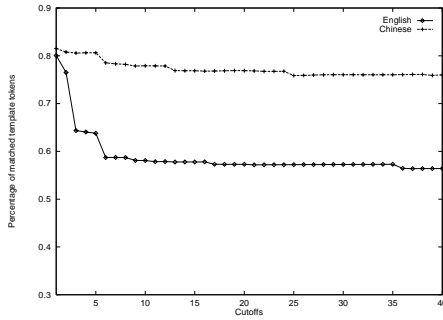


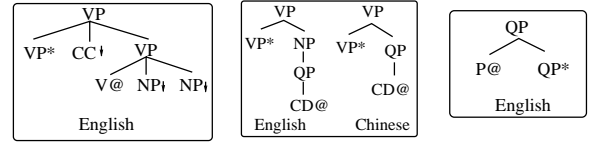
Figure 6: The percentages of matched template tokens in the English and Chinese Treebanks with various threshold values

4.4 Unmatched Templates

Our experiments (see Table 4 and 6) show that the percentages of unmatched template tokens in three Treebanks range from 14.4% to 52.4%, depending on the language pairs and the threshold value. Given a language pair, there are various reasons why a template appears in one Treebank grammar, but not in the other. We divide those unmatched templates into two categories: spuriously unmatched templates and truly unmatched templates.

Spuriously unmatched templates *Spuriously* unmatched templates are those that either should have found a matched template in the other grammar or should not have been created by LexTract in the first place if the Treebanks had been complete, uniformly annotated, and error-free. A spuriously unmatched template might exist because of one of the following reasons:

(S1) Treebank coverage: The template is linguistically sound in both languages, and, therefore, should belong to the grammars for these languages. However, the template appears in only one Treebank grammar because the other Treebank is too small to include such a template. Figure 7(S1) shows a template that is valid for both English and Chinese, but it appears only in the English Treebank, not in the Chinese Treebank.



(S1) Treebank coverage (S2) annotation difference (S3) annotation error

Figure 7: Spuriously unmatched templates

(S2) Annotation differences: Treebanks may choose different annotations for the same constructions; consequently, the templates for those constructions look different. Figure 7(S2) shows the templates used in English and Chinese for a VP such as “*surged 7 (dollars)*”. In the template for English, the *QP* projects to an *NP*, but in the template for Chinese, it does not.

(S3) Treebank annotation errors: A template in a Treebank may result from annotation errors in that Treebank. If no corresponding mistakes are made in the other Treebank, the template in the first Treebank will not match any template in the second Treebank. For instance, in the English Treebank the adverb *about* in the sentence *About 50 people showed up* is often mis-tagged as a preposition, resulting in the template in Figure 7(S3). Not surprisingly, that template does not match any template in the Chinese Treebank.

Truly unmatched templates A *truly* unmatched template is a template that does not match any template in the other Treebank even if we assume both Treebanks are perfectly annotated. Here, we list three reasons why a truly unmatched template might exist.

(T1) Word order: The word order determines the positions of dependents with respect to their heads. If two languages have different word orders, the templates that include dependents of a head are likely to look different. For example, Figure 8(T1) shows the templates for transitive verbs in Chinese and Korean grammars. They do not match because of the different positions of the object of the verb.

(T2) Unique tags: For each pair of languages, some Part-of-speech tags and syntactic tags may appear in only one

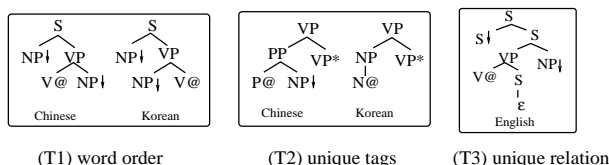


Figure 8: Truly unmatched templates

language. Therefore, the templates with those tags will not match any templates in the other language. For instance, in Korean the counterparts of preposition phrases in English and Chinese are noun phrases (with postpositions attaching to nouns), as shown in the right figure in Figure 8(T2); therefore, the templates with PP in Chinese, such as the left one in Figure 8(T2), do not match any template in Korean.

(T3) Unique syntactic relations: Some syntactic relations may be present in only one of the pair of languages being compared. For instance, the template in Figure 8(T3) is used for the sentence such as “*You should go,*” said John, where the subject of the verb *said* appears after the verb. No such template exists in Chinese.

	S1	S2	S3	T1	T2	T3	total
type(#)	1	70	53	22	99	65	310
token(%)	0.0	3.2	0.2	0.7	12.3	2.1	18.5

Table 7: The distribution of the Chinese templates that do not match any English templates

So far, we have listed six possible reasons for unmatched templates. We have manually classified templates that appear in the Chinese grammar, but not in the English grammar.⁵ The results are shown in Table 7. The table shows that for the Chinese-English pair, the main reason for unmatched templates is (T2); that is, the Chinese Treebank has tags for particles (such as aspect markers and sentence-ending particles), which do not exist in English. For other language pairs, the distribution of unmatched templates may be very different. For instance, Table 4 indicates that the English grammar covers 85.6% of the template tokens in the Korean Treebank. If we ignore the word order in the templates, that percentage increases from 85.6% to 97.2%. In other words, the majority of the template tokens that appear in the Korean Treebank, but not in the English Treebank, are due to the word order difference in the two languages. Note that the word order difference only accounts for a small fraction of the unmatched templates in the Chinese-English pair (see the fifth column in Table 7). This contrast is not surprising considering that English and Chinese are predominantly head-initial, whereas Korean is head-final.

5 Conclusion

We have presented a method of quantitatively comparing grammars extracted from Treebanks. Our experimental re-

⁵For this experiment, we used all the templates in the grammars; that is, we did not throw away low frequency templates.

sults show a high proportion of easily inter-mappable structures, providing support for the Universal Grammar hypothesis. We have also described a number of reasons why a particular template does not match any templates in the other languages and tested the effect of word order on matching percentages.

There are two natural extensions of this work. First, running an alignment algorithm on parallel bracketed corpora would produce word-to-word mappings. Given such word-to-word mappings and our template matching algorithm, we can automatically create lexicalized *etree-to-etree* mappings, which can be used for semi-automatic transfer lexicon construction. Second, LexTract can build derivation trees for each sentence in the corpora. By comparing derivation trees for parallel sentences in two languages, instances of structural divergences [Dorr, 1993] can be automatically detected.

References

- [Comrie, 1987] Bernard Comrie. *The World's Major Languages*. Oxford University Press, New York, 1987.
- [Dorr, 1993] B. J. Dorr. *Machine Translation: a View from the Lexicon*. MIT Press, Boston, Mass., 1993.
- [Han et al., 2001] Chunghye Han, Na-Rae Han, and Eon-Suk Ko. Bracketing Guidelines for the Penn Korean Treebank (forthcoming), 2001. www.cis.upenn.edu/~xtag/koreantag.
- [Joshi and Schabes, 1997] Aravind Joshi and Yves Schabes. Tree Adjoining Grammars. In A. Salomaa and G. Rosenberg, editors, *Handbook of Formal Languages and Automata*. Springer-Verlag, Herdelberg, 1997.
- [Joshi et al., 1975] Aravind K. Joshi, L. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 1975.
- [Marcus et al., 1993] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics*, 1993.
- [Sarkar, 2001] Anoop Sarkar. Applying Co-Training Methods to Statistical Parsing. In *Proc. of the 2nd NAACL*, 2001.
- [Xia et al., 2000a] Fei Xia, Martha Palmer, and Aravind Joshi. A Uniform Method of Grammar Extraction and its Applications. In *Proc. of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, 2000.
- [Xia et al., 2000b] Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Shizhe Huang, Tony Kroch, and Mitch Marcus. Developing Guidelines and Ensuring Consistency for Chinese Text Annotation. In *Proc. of the 2nd International Conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece, 2000.
- [Xia, 1999] Fei Xia. Extracting Tree Adjoining Grammars from Bracketed Corpora. In *Proc. of 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Beijing, China, 1999.