The Workshop Programme

9:00-9:30	Rebecca Hwa, Philip Resnik, Amy Weinberg	*Breaking the Resource Bottleneck for Multilingual Parsing*
9:30-10:00	Aoife Cahill, Mairead McCarthy, Josef van Genabith, Andy Way	*Automatic Annotation of the Penn-Treebank with LFG F-Structure Information*
10:00-10:30	Kiril Simov, Milen Kouylekov, Alexander Simov	*Incremental Specialization of an HPSG-Based Annotation Scheme*
10:30-11:00	Bernd Bohnet, Stefan Klatt, Leo Wanner	*A Bootstrapping Approach to Automatic Annotation of Functional Information to Adjectives with an Application to German*
11:00-11:30	Coffee break	
11:30-12:00	Adam Lopez, Mike Nossal, Rebecca Hwa, Philip Resnik	*Word-Level Alignment for Multilingual Resource Acquisition*
12:00-12:30	Necip Fazil Ayan, Bonnie J. Dorr	*Generating a Parsing Lexicon from an LCS-Based Lexicon*
12:30-13:00	Alberto Lavelli, Bernardo Magnini, Fabrizio Sebastiani	*Building Thematic Lexical Resources by Bootstrapping and Machine Learning*
13:00-14:30	Lunch break	
14:30-15:00	Anja Belz	*Learning Grammars for Noun Phrase Extraction by Partition Search*
15:00-15:30	Fermín Moscoso del Prado Martín, Magnus Sahlgren	*An Integration of Vector-Based Semantic Analysis and Simple Recurrent Networks for the Automatic Acquisition of Lexical Representations from Unlabeled Corpora*
15:30-16:00	Pavel Kveton, Karel Oliva	*Detection of Errors in Part-Of-Speech Tagged Corpora by Bootstrapping Generalized Negative n-Grams*
16:00-16:30	Coffee break	
16:30-17:00	Rayid Ghani, Rosie Jones	*A Comparison of Efficacy and Assumptions of Bootstrapping Algorithms for Training Information Extraction Systems*
17:00-17:30	Marisa Jiménez	*Using Decision Trees to Predict Human Nouns in Spanish Parsed Text*
17:30-18:00	Laura Alonso, Irene Castellón, Lluís Padró	*X-Tractor: A Tool for Extracting Discourse Markers*

Workshop Organisers

Alessandro Lenci	Università di Pisa, Italy
Simonetta Montemagni	Istituto di Linguistica Computazionale - CNR, Italy
Vito Pirrelli	Istituto di Linguistica Computazionale - CNR, Italy

Workshop Programme Committee

Harald Baayen	Max Planck Institute for Psycholinguistics - Nijmegen, The Netherlands
Rens Bod	University of Amsterdam, Holland
Michael R. Brent	Washington University, USA
Nicoletta Calzolari	Istituto di Linguistica Computazionale - CNR, Italy
Jean-Pierre Chanod	Xerox Research Centre Europe, Grenoble, France
Walter Daelemans	University of Antwerp, Belgium
Dekang Lin	University of Alberta, Edmonton, Canada
Horacio Rodriguez	Universidad Politecnica de Catalunya
Fabrizio Sebastiani	Istituto per l'Elaborazione dell'Informazione - CNR, Italy
Lucy Vanderwende	Microsoft Research, Redmond, USA
François Yvon	Ecole Nationale Superieure des Telecommunications, Paris Frances
Menno van Zaanen	University of Amsterdam, The Netherlands

Table of Contents

Author Index

Preface

Provision of large-scale language resources, such as tagged corpora, lexicons and repositories of pre-classified text documents, is a crucial key to steady progress in an extremely wide spectrum of research, technological and business areas in the HLT sector. The continuously changing demands for language-specific and application-dependent annotated data (*e.g.* at the syntactic or at the semantic level), indispensable for design validation and efficient software prototyping, however, are daily confronted by the *resource bottleneck*. Handcrafted resources are often too costly and time-consuming to be produced at a sustainable pace, and, in some cases, they even exceed the limits of human conscious awareness and descriptive capability. The problem is even more acutely felt for low-resource languages, since the early stages of language resource development often require gathering considerable momentum both in terms of know-how and level of funding, of the order of magnitude normally deployed by large national projects.

Possible ways to circumvent, or at least minimise, these problems come from the literature on automatic knowledge acquisition and, more generally, from the machine-learning community. Of late, a number of machine learning algorithms have proved to fare reasonably well in the task of incrementally bootstrapping newly annotated data from a comparatively small sample of already annotated resources. Another promising route consists in automatically tracking down recurrent knowledge patterns in relatively unstructured or implicit information sources (such as free texts or machine readable dictionaries) for this information to be moulded into explicit representation structures (e.g. subcategorization frames, syntactic-semantic templates, ontology hierarchies etc.). In a similar vein, several strategies have been investigated aimed at merging or integrating structured information sources into a unitary comprehensive resource, or at customising general-purpose knowledge-bases for them to be of use in more technical domains. Finally, the growing availability of multi-lingual parallel resources has prompted the idea of using a high-resource language (generally English) to fertilize a low-resource language.

We believe that all these attempts at bootstrapping annotated language data are not only of practical interest, but also point to a bunch of germane theoretical issues. Gaining insights into the deep interrelation between representation and acquisition issues is likely to have significant repercussions on the way linguistic resources will be designed, developed and used for applications in the years to come. As the two aspects of knowledge representation and acquisition are profoundly interrelated, progress on both fronts can only be achieved, in our view of things, through a full appreciation of this deep interdependency.

The papers contained in this volume (13 out of 20 submissions) significantly confirm this general view. They focus on a variety of bootstrapping techniques to show their full potential for the provision of annotated language data. In particular, three areas of investigation are dealt with in some detail (often concurrently in the same paper). At the level of corpus annotation, parsers and annotated texts are increasingly used as a tightly integrated resource. Parsing robustness is no longer an end in itself but forms part of a virtuous incremental circle whereby finer grained, more accurate or simply more explicit levels of text analysis are built probabilistically on the basis of either under-specified or possibly more compact annotations. This process proves to be able to provide increasingly richer annotated data and sheds considerable light on the issue of inter-annotation translatability. At the level of lexicon design and building, a lot of effort is being put into merging complementary levels of language information (e.g. semantic and syntax, or semantic and morphology) to produce better lexical repositories for parsing and better computational models of the internalised lexical competence of a speaker. This strikes us as an extremely promising route, bound to throw in sharper relief the importance of simultaneously dealing with more information levels in parsing real texts at the level of accuracy required by HLT applications. Emphasis on the use of available machine learning technology for dealing with the novel challenges of HLT applications is the third research prong of this volume. Current and future needs for information extraction, classification and management appear to impose novel requirements on text processing and create novel tasks in the HLT sector. Once more, machine-learning approaches play an important role here. Perhaps even more significantly, these tasks provide, in turn, a key to a deeper understanding of the implicit assumptions underlying different machine-learning techniques.

We would like to thank all the authors who showed their interest by submitting papers to the workshop. We would also like to thank the members of the programme committee who kindly contributed to the reviewing process and the scientific and programme committees of LREC 2002.

Alessandro Lenci
<alessandro.lenci@ilc.cnr.it>
Università di Pisa, Italy

Simonetta Montemagni
<simonetta.montemagni@ilc.cnr.it>
Vito Pirrelli
<vito.pirrelli@ilc.cnr.it>
Istituto di Linguistica Computazionale - CNR, Italy

Breaking the Resource Bottleneck for Multilingual Parsing

Rebecca Hwa[1], Philip Resnik[1,2], and Amy Weinberg[1,2]

Institute for Advanced Computer Studies[1]
Department of Linguistics[2]
University of Maryland, College Park, MD 20742
{hwa, resnik, weinberg}@umiacs.umd.edu

Abstract

We propose a framework that enables the acquisition of annotation-heavy resources such as syntactic dependency tree corpora for low-resource languages by importing linguistic annotations from high-quality English resources. We present a large-scale experiment showing that Chinese dependency trees can be induced by using an English parser, a word alignment package, and a large corpus of sentence-aligned bilingual text. As a part of the experiment, we evaluate the quality of a Chinese parser trained on the induced dependency treebank. We find that a parser trained in this manner out-performs some simple baselines inspite of the noise in the induced treebank. The results suggest that projecting syntactic structures from English is a viable option for acquiring annotated syntactic structures quickly and cheaply. We expect the quality of the induced treebank to improve when more sophisticated filtering and error-correction techniques are applied.

1 Introduction

There is a substantial disparity between the quality of state of the art parsers available for English and those for other languages. English parsers such as those of Collins (1997) and Charniak (1999) were trained on hand annotated corpora such as the Penn Treebank Project (Marcus et al., 1993). However, experience has shown us that building hand-crafted treebanks from scratch is too time-consuming to be repeated for every language of interest. This bad news can be mitigated by leveraging English annotations to automatically acquired annotations for new languages. Recent work by Yarowsky and Ngai (2001) has shown that this type of transfer is possible for inducing part-of-speech tags for Chinese. In this paper, we explore the application of this technique to the more complex problem of inducing Chinese dependency trees.

The input to our system is a collection of sentence-aligned bilingual text (i.e., pairs of sentences that are translations of each other). Each English sentence is parsed using a high-quality English parser. For each pair of sentences, word alignment is performed using statistical MT models (Brown et al., 1990; Al-Onaizan et al., 1999). The alignment then anchors the projection of the English tree to the Chinese side (see Figure 1).

This paper presents an initial large-scale experiment, investigating the feasibility of inducing a Chinese dependency treebank using our projection algorithm and of training a parser on the resulting treebank. Due to the compounded errors of various components of the system, the induced Chinese dependency treebank is rather noisy. Applying filtering heuristics to the treebank improves its quality enough such that the parser trained on it out-performs some simple baselines. While the parser's performance is still significantly less than that of a parser trained on a clean, fully annotated (Chinese) treebank, this study suggests that projecting syntactic structures from English is viable for acquiring annotated syntactic structures quickly and cheaply.

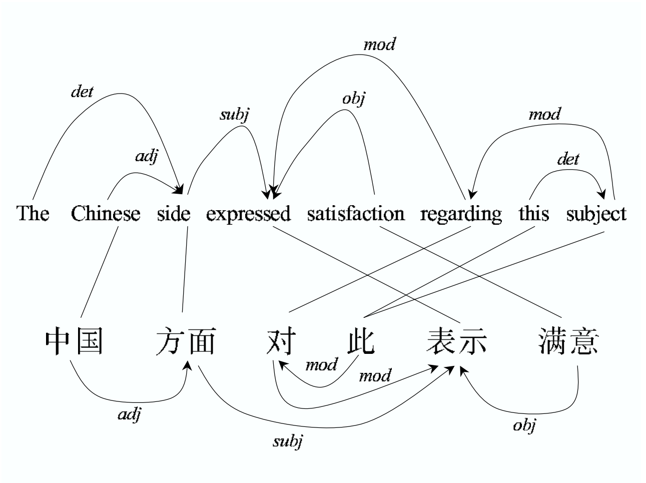

mod
det
subj
obj
mod
adj
det
The Chinese side expressed satisfaction regarding this subject
中国 方面 对 此 表示 满意
mod
mod
adj
subj
obj

Figure 1: Given an English dependency parse tree and a set of word alignments, we infer the syntactic structure on the Chinese side via projection from its English counterpart.

2 Overview of the Algorithm

Our approach requires three resources. First, we need a sizable, sentence-aligned bilingual text as training corpus. In our experiment, we use a bilingual text of English and Chinese news articles. In Section 5 we discuss other ways in which bilingual text can be acquired and sentence aligned. Second, we require dependency parses of the English text. Our choice of dependency representation is motivated in Section 2.1. Third, word alignments are needed to relate the sentence pair on the lexical level. In this paper, we use alignments produced as a side-effect of training a statistical translation model (Brown et al., 1990; Al-Onaizan et al., 1999).

Given these resources, our system behaves as follows: for each sentence pair (E, C) in the bilingual text, the English sentence E is parsed and converted into a dependency representation. Next, word alignment is performed for the sentence pair. Finally, the English dependency analysis is projected across the word alignment to the Chinese side according to our *Direct Projection Algorithm*, which we outline in section 2.2.

2.1 Dependency Representations as Transfer Medium

Dependency relationships specify asymmetric binary relations between two surface words: a *head* and its *modifier*. For example, in the sentence from Figure 1, *"The Chinese side expressed satisfaction regarding this subject,"* the word *side* modifies the head word *expressed*. The dependency links may optionally be annotated with information specifying grammatical relations between constituents such as *subject*, *object*, *modifier*, etc. In our example, the link between *side* and *expressed* is labeled as *subj*, indicating that the constituent *The Chinese side* is the subject of the verb *expressed*. In this section, we argue that dependency representation is right for our projection framework because it captures both structural and lexical relationships between words that are not string local; because it overcomes some of the shortcomings of evaluating against the phrase structure representation; and because it is language independent with respect to word order variations.

Syntactic analysis in terms of phrase structure has been the dominant paradigm in natural language processing, starting from early context-free grammars and continuing up to present-day stochastic formalisms. It is preferable over models that make Markov assumptions restricting interactions among words to those that occur within the window of an n-gram. Phrase structure formalisms provide a level of representation that allows significant constraint to occur between grammatical categories that are not string-local. These categories become *local* at the phrase structure level. For example, consider the following sentence from the Brown Corpus:

> *The largest hurdle the Republicans would have to face is a state law which says that before making a first race, one of two alternative courses must be taken.*

The relationship between *hurdle* and *is* exists over a long string-distance, owing to an embedded relative clause, and, similarly, *Republicans* and *face* are separated in the string by a sequence of auxiliaries and the infinitival *to*. As a result, the relationships represented in the sentence are not captured well by any n-gram model with tractable n. In contrast, the relationship between the subject NP and the predicate is easily encoded locally within a context-free rule such as S $\rightarrow$ NP VP.

To take full advantage of such relationships in models based on phrase structure, however, it is necessary to *lexicalize* the grammar formalism, so that lexically-based constraints are also localized within grammar rules. By incorporating lexical content into phrase structure rules (e.g., S(is) $\rightarrow$ NP(hurdle) VP(is)), lexicalized grammar formalisms make it possible to capture syntactic constraints such as as number agreement (e.g. the low probability of S(are) $\rightarrow$ NP(hurdle) VP(are)) as well as semantic constraints (e.g. the reasonably high probability of S(face) $\rightarrow$ NP(Republicans) VP(face)). Work taking advantage of this insight (e.g. Collins (1997; Charniak (1999)) has defined the breakthroughs leading to the current state of the art in broad-coverage parsing. Implicitly or sometimes explicitly (as in the work of Collins), what gives lexicalized context-free representations their power is the ability to probabilistically model the syntactic *dependency* relationships between words in the structure.

Moreover, dependency analysis evaluation avoids some of the shortcomings of constituency analysis evaluation (Lin, 1995; Carroll et al., 1999). Standard constituency parsing metrics compare the phrase boundaries specified by the gold standard to that of the candidate analysis. They also evaluate whether conditions on well formed trees (such as a ban on crossing branches) are respected by the candidate. However, as Lin (1995) notes, since branching structure is not directly tied to semantic interpretation, it is unclear how to interpret missing, spurious, or crossing branches. On the other hand, it is apparent that syntactic dependencies, more so than syntactic constituents, are closely tied to the who-did-what-to-whom relationships of language. Indeed, work in lexical semantics relating syntactic representations to thematic relationships such as *agent*, *theme*, *beneficiary*, has focused primarily on syntactic dependencies rather than on phrasal constituents (Baker, 1997). Since semantic dependencies form a superset based on syntactic dependencies, we are better able to gauge how likely a representation is to be interpretable, by measuring the percentage of correct dependencies.

Finally, dependency structures firmly separate precedence from dominance relations, such that word order variation between languages becomes less of a problem than in constituency trees. For example, the relative string order of a series of modifiers of a head is irrelevant in the dependency representation. All are modifiers. By contrast, a constituency tree may require a stacked structure that would not translate well if the word order were reversed in another language. In other words, dependency structures are more likely to respect a homomorphism.

These observations suggest that dependencies may be a better choice for syntactic projection across languages than phrasal constituents. To the extent that this assumption is correct, we should be able to use word alignments as a bridge between English and another language, retaining some level of confidence that if dependencies are projected across the alignment they will be correct for the new language. Experimental results from our previous work (Hwa et al., 2002), have indicated that while the assumption does not always hold true, syntactic analyses projected from English to Chinese can, in principle, yield Chinese analyses that are nearly 70% accurate (in terms of unlabeled dependencies) after application of a set of linguistically principled rules.[1]

2.2 The Direct Projection Algorithm

Our approach is based on the intuitive idea of a direct projection of dependency structures. We now describe our

[1]The experiment was performed under idealized settings, projecting human annotated English dependency analyses using human annotated word alignments.

projection algorithm in more detail. Given sentence pair (E, C), where $E = e_1, \ldots, e_n$ and $C = c_1, \ldots, c_m$, syntactic relations (denoted as $R(x, y)$) are projected from English for the following situations:

- **one-to-one** if e_i is aligned with a unique c_x and e_j is aligned with a unique c_y, if $R(e_i, e_j)$, conclude $R(c_x, c_y)$.
- **unaligned (English)** if e_j is not aligned with any word in C, then create a new empty word c_y such that for any e_i aligned with a unique c_x, $R(e_i, e_j) \Rightarrow R(c_x, c_y)$ and $R(e_j, e_i) \Rightarrow R(c_y, c_x)$.
- **one-to-many** if $e_i$is aligned with $c_x, \ldots, c_y$, then create a new empty word c_z such that c_z is the parent of $c_x, \ldots, c_y$ and set e_i to align to c_z instead. We called this a *Multiply-Aligned Component*, or MAC.
- **many-to-one** if $e_i, \ldots, e_j$ are all uniquely aligned to c_x, then delete all alignments between $e_k (i \leq k \leq j)$ and c_x except for the head of $e_i, \ldots, e_j$.

The **many-to-many** case is decomposed into a two-step process: first perform one-to-many, then perform many-to-one. In the cases of unaligned Chinese words, they are left out of the projected syntactic tree. The asymmetry of the treatment of **one-to-many** and **many-to-one** and of the unaligned words for the two languages arises from the asymmetric nature of the projection.

2.2.1 Post-Projection Transformation

The Direct Projection Algorithm by itself does not produce good dependency trees because it does not properly handle structural projection for the more complex cases when the alignment is not one-to-one. Therefore, we apply a small set of linguistically motivated rules to correct the projected trees as a post-hoc process. It is clearly an advantage to limit the correction rules to those that can apply generally, across many construction types. Wanting to avoid unending language-specific rule tweaking, we strictly limited the possible rules. Rules were permitted to refer only to closed class items, to parts of speech projected from the English analysis, or to easily enumerated lexical categories (e.g. {*dollar*, RMB, $, yen}). The majority of rule patterns are variations on the same solution to the same problem. Viewing the problem from a higher level of linguistic abstraction made it possible to find all the relevant cases in a short time and express the solution compactly; in all, fewer than twenty rules were written, and the analysis, rule writing, and verification of their correctness using the data set took a few days.

Here are two examples of the rules we developed; see (Hwa et al., 2002) for fuller discussion.

Rule for noun modification:

- If $c_x, \ldots, c_y$ are a set of Chinese words aligned to an English noun, replace the empty node introduced in the Direct Projection Algorithm by promoting the last word c_y to its place with $c_x, \ldots, c_{y-1}$ as dependents.

Rule for aspectual markers:

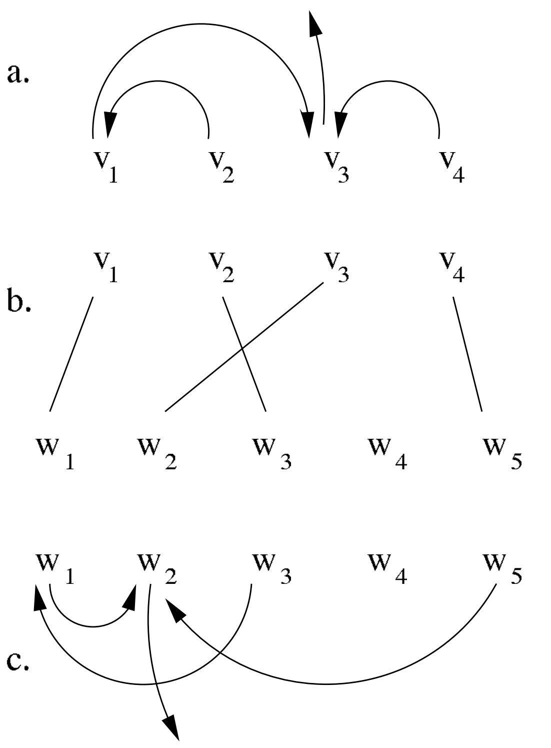


Figure 2: The direct projection of the dependency parse for $v_1 \ldots v_4$ (Figure 2a) across the word alignment (Figure 2b) results in cross dependency relationships for the link between w_1 and w_3 and the link between w_2 and w_5; and it leaves word w_4 unattached to the projected dependency tree (Figure 2c).

- If $c_x, \ldots, c_y$, a sequence of Chinese words aligned with English verbs, is followed by c_a, an aspect marker, make c_a into a modifier of the last verb c_y.

2.2.2 Remaining Shortcomings of the Direct Projection Algorithm

Although the majority of the projected trees are significantly improved, the post-projection transformation rules still do not adequately address some major deficiencies of the Direct Projection Algorithm. The algorithm does not ensure that the projected structure is indeed a well-formed structure. Thus, when given unconstrained word alignment outputs, the projected structure may contain errors such as crossing dependencies (see Figure 2). Moreover, due to the asymmetry of the algorithm, the syntactic role of unaligned foreign words cannot be inferred. The post-projection transformation rules address this problem to some extent by incorporating unaligned function words back into the parse, but an intelligent treatment of the open class of unaligned words remains a challenge of this projection approach. Furthermore, the algorithm does not address complex translation divergences (Dorr, 1993), such as the head-swapping phenomenon (in which the direction of the head-modifier dependency is reversed in the foreign language). Lopez et al. (2002) describe an alternative to the direct projection approach that addresses some of these problems.

3 Experimental Setup

Our previous results have shown that, given good English parses and clean alignments to Chinese translations, the direct projection approach from English to Chinese (together with post-processing) can lead to Chinese annotations that are substantially correct; unlabeled precision/recall on projected dependencies approaches 70% (Hwa et al., 2002). While this demonstrates that the approach holds promise in automatically inducing syntactic treebanks of reasonable quality, it is not clear how much degradation occurs when using imperfect English parsers and imperfect word alignment models. That question is our focus in this paper. We report a full-scale experiment on English and Chinese sentence pairs, evaluating the entire framework under the realistic settings of imperfect bilingual data and error-prone parsers and alignment models (see Section 3.1). Once a Chinese dependency treebank is induced, we use it to train a Chinese parser in a manner similar to that of Collins (1999). The trained parser is then evaluated on unseen test sentences taken from the Chinese Treebank (Xia et al., 2000) and compared with two baselines and an upper bound.

3.1 Resources

We use about 56,000 sentence pairs from the Hong Kong News (HKNews) corpus as our bilingual text. The data have been automatically sentence aligned and the Chinese words have been automatically segmented.[2] To parse the English sentences, we use a lexicalized statistical parser trained on the Wall Street Journal corpus (Collins, 1997).[3] To obtain word alignments for all sentence pairs, we train an off-the-shelf statistical translation model, GIZA++ (Al-Onaizan et al., 1999), using the HKNews bilingual text. Given these resources, the direction projection algorithm and the post-projection transformation process are then used to induce dependency trees for the Chinese sentences in the HKNews corpus.

3.2 Evaluation of the Induced Treebank

Because of its size, we do not directly assess the quality of the induced treebank. Instead, we evaluate the Chinese parser trained from it. To the extent that the trained parser outputs reasonable structures on unseen test sentences, it indicates that the induced treebank is a useful resource. To evaluate the quality of the trained parser, we compare it to two simple baseline dependency analyses: *always modify the previous word*, and *always modify the next word*. As an upper bound, we have also trained the same parser with clean, hand-annotated trees from the Penn Chinese Treebank (ChTB). We constructed a development set consisting of 124 sentences and a test set consisting of 88 sentences taken from the Chinese Treebank; all sentences are of 40 words or less. The remaining approximately 3800 Chinese Treebank sentences are converted into their dependency representation (similar to the algorithm described in Section 2 of the paper by Xia and Palmer (2001)) and used as training data for the upper-bound parser. We evaluate the trained parser by comparing its output (dependency) parse trees for the unseen test sentences against the human-annotated gold standard parse trees (also converted to dependency representation). The metrics used are the precision and recall scores on the unlabeled dependency relations. A parser produced dependency link is considered "correct" if the same head-modifier relationship exists in the gold standard; the dependency label does not need to match. Punctuations are not scored.

4 Results and Discussions

Tables 1 and 2 show performance comparisons for our automatic projection approach as compared to the lower and upper bounds. As one might expect, the quality of the treebank induced under the real-world constraints of imperfect data and components is noticeably worse than one induced using clean English parses and perfect word alignments. The Direct Projection Algorithm and its associated post-projection transformation rules are not fault-tolerant enough to recover from the compounding errors of the parser and alignment model. Without further processing, the projected treebank would contain too much noise to be useful for training a parser. Therefore, our attentions turn to filtering heuristics for poorly induced dependency trees.

We found that the most unreliable component is the word alignment model. A cursory inspection of the alignment output (for the HKNews corpus) shows that, for many sentences, the majority of the English words remain unaligned; and that often, an unusually high number of Chinese words (e.g, five or greater) are aligned to the same English word. The poor alignment output may have many causes: in particular, the sentence pair input to the alignment model is imperfect, and the alignment model does not perform well for language pairs with very dissimilar word-order patterns.

This suggests that performance might improve if we filter out sentence pairs that are known to be poorly aligned. To filter out dependency trees projected from dubious word alignments, we have devised several simple heuristics. First, we removed those sentences for which more than 30% of the English words were not aligned to any Chinese word ($EnoC \leq 0.3$). The figure 30% is empirically determined, based on the trained parser's performance on the development set. As shown in the first row of Table 1, the parser trained on the filtered treebank does outperform the modify-next baseline; however, the corpus size has been drastically cut-down from around 56,000 to less than 8,000. The second filter we apply to the corpus is to remove sentences in which the size of a multiply aligned component is greater than three (MAC > 3); that is, when more than three Chinese words are aligned to the same English word. The MAC value of 3 was also determined empirically using development data. The second line of Table 1 shows that training the parser on the induced treebank filtered by both heuristics leads to further improvement. Finally, we return to the crossing-dependency problem alluded to earlier in section 2.2.2. While we do not correct the crossing depen-

[2] We are grateful to Stefan Vogel of CMU for his assistance with this corpus.

[3] The executable of the parser is freely available at ftp://ftp.cis.upenn.edu/pub/mcollins/misc.

Method	Corpus Size	Precision & Recall
EnoC	7689	37.4
EnoC+MAC	5525	42.1
EnoC+MAC+NoCross	5284	42.9
Modify Prev (Baseline)	–	14.0
Modify Next (Baseline)	–	32.2

Table 1: The parser's performance on the development set (%) when the training corpus has been filtered with the following heuristics: remove sentences if too many English words have no Chinese translations (EnoC); remove sentences if too many Chinese words are aligned to one English word (MAC); remove sentences that violate many crossing-dependency constraints (NoCross).

dencies in this work, we remove sentences with the most egregious crossing-dependency violations in their analyses. Our experiments with development data suggested that a sentence should be filtered out if more than 40% of its dependency links violate the no-crossing constraint. The combination of the three filters improved the induced treebank so that a parser trained on the treebank outperforms the simple baselines; however, the draconian filters also reduced the corpus from 56,000 sentences to slightly over 5,000.

Table 2 shows the trained parser's performance on a separate test set. As before, it is compared with two baselines; and as an upper bound, we train the same parser on a clean, manually created treebank.[4] Similar to the outcome of the development set, the trained parser performs better than the baseline, but it still cannot compete with a parser trained on a clean corpus. It is interesting to note that after our current filtering techniques, the sizes of the induced treebank is comparable to the clean one. However, our method of treebank acquisition is not constrained by the laborious manual annotation process; therefore it would be easy for us to obtain a much larger bilingual corpus as a starting point, as discussed below. We conjecture that the size of the corpus will help offset the effect of the noise, as will more sophisticated sampling techniques that exclude the noisiest data.

5 Conclusion and Future Work

In this paper, we have described our framework for acquiring Chinese dependency treebanks by bootstrapping from existing linguistic resources for English. We have explicitly discussed the assumptions made and the resources required in order for our algorithm to work. An ambitious full-scale experiment using real-world data was performed to investigate the feasibility of our approach. Our results suggest that treebank acquisition through projection is indeed possible; however reducing the noise in the induced treebank is a major challenge.

This finding points us to several directions for further research. One clear avenue is to obtain larger bilingual texts, so that more data remain even when noisy sentence pairs have been filtered out. Work on mining the Web for bilingual text, such as STRAND (Resnik, 1999), BITS (Ma and Liberman, 1999), and PTMiner (Nie et al., 1999), show significant promise in this regard. Once parallel Web pages are obtained, it is possible to obtain sentence- or segment-level alignments either via alignment of HTML markup (Resnik, 1998) or via more sophisticated sentence-alignment techniques (Melamed, 1998).

Beyond simply taking a "more is better" approach to data acquisition, one way to reduce the noise in the induced treebank is to lower the error rates of the individual components in our projection framework. Of these, improving the word alignment model would benefit the overall system the most. We are actively developing alternative word alignment models that is sensitive to this syntactic projection framework (Lopez et al., 2002). Moreover, as we have shown in this study, filtering techniques that identify and remove malformed trees can help reducing noise; however, aggressive filtering alone is likely to result in over-filtering. To render nearly 90% of the bilingual text useless places too heavy a burden on even the best Web mining techniques. We are experimenting with filtering strategies that attempt to localize the potentially problematic parts of a syntactic tree so that the rest can still contribute to the training corpus. In addition, we are continuing to work on the post-projection transformation the process to improve the quality of the projected trees.

[4]The upper-bound parser's performance is on par with that of the state of the art constituency parsers trained on the Chinese Treebank, e.g. (Bikel and Chiang, 2000).

6 Acknowledgments

This work has been supported, in part, by ONR MURI Contract FCPO.810548265, NSA RD-02-5700, DARPA/ITO Cooperative Agreement N66001002891O and Mitre Contract 010418-7712. The authors would like to thank Franz Josef Och for his help with using the GIZA++ translation model; and Adam Lopez and Mike Nossal for helpful discussions and comments on this paper.

7 References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, I. Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation. Technical report, JHU. citeseer.nj.nec.com/al-onaizan99statistical.html.

Mark C. Baker, 1997. *Thematic Roles and Syntactic Structure*, pages 73–137. Kluwer.

Daniel Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, pages 1–6.

Method	Corpus	Size	Precision & Recall
Modify Prev (Baseline)	–	–	13.5
Modify Next (Baseline)	–	–	35.7
Stat. Parser	Induced HKNews	5284	42.3
Stat. Parser (Upper-bound)	Clean ChTB	3870	75.6

Table 2: A comparison of the parsers' performance against lower and upper bounds on the test set (%).

Peter F. Brown, John Cocke, Stephen A. DellaPietra, Vincent J. DellaPietra, Frederick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, June.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *LINC-99 workshop at the 9th Conference of the EACL*, June.

Eugene Charniak. 1999. A maximum-entropy inspired parser. Technical Report CS-99-12, Brown University.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, pages 16–23, Madrid, Spain.

Michael Collins. 1999. A statistical parser for czech. In *Proceedings of the 37th Annual Meeting of the ACL*, College Park, Maryland.

Bonnie J. Dorr. 1993. *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, MA.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the ACL*. To appear.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the IJCAI-95*, pages 1420–1425.

Adam Lopez, Michael Nossal, Rebecca Hwa, and Philip Resnik. 2002. Word-level alignment for multilingual resource acquisition. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*. To appear.

Xiaoyi Ma and Mark Liberman. 1999. Bits: A method for bilingual text search over the web. In *Machine Translation Summit VII*.

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

I. Dan Melamed. 1998. *Empirical Methods for Exploiting Parallel Texts*. Ph.D. thesis, University of Pennsylvania, May.

J. Nie, M. Simard, P. Isabelle, and R. Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining parallel texts from the web. In *Proceedings of the ACM SIGIR Conference*.

Philip Resnik. 1998. Parallel strands: A preliminary investigation into mining the Web for bilingual text. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas, AMTA-98, in Lecture Notes in Artificial Intelligence, 1529*, Langhorne, PA, October 28-31.

Philip Resnik. 1999. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the ACL*, June.

Fei Xia and Martha Palmer. 2001. Converting dependency structures to phrase structures. In *Proc. of the HLT Conference*, March.

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

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proc. of NAACL-2001*, pages 200–207.

Automatic Annotation of the Penn-Treebank with LFG F-Structure Information

Aoife Cahill, Mairead McCarthy, Josef van Genabith, Andy Way

School of Computer Applications, Dublin City University
Dublin 9, Ireland
{acahill, mcarthy, josef, away}@computing.dcu.ie

Abstract

Lexical-Functional Grammar f-structures are abstract syntactic representations approximating basic predicate-argument structure. Treebanks annotated with f-structure information are required as training resources for stochastic versions of unification and constraint-based grammars and for the automatic extraction of such resources. In a number of papers (Frank, 2000; Sadler, van Genabith and Way, 2000) have developed methods for automatically annotating treebank resources with f-structure information. However, to date, these methods have only been applied to treebank fragments of the order of a few hundred trees. In the present paper we present a new method that scales and has been applied to a complete treebank, in our case the WSJ section of Penn-II (Marcus et al, 1994), with more than 1,000,000 words in about 50,000 sentences.

1. Introduction

Lexical-Functional Grammar f-structures (Kaplan and Bresnan, 1982; Bresnan, 2001) are abstract syntactic representations approximating basic predicate-argument structure (van Genabith and Crouch, 1996). Treebanks annotated with f-structure information are required as training resources for stochastic versions of unification and constraint-based grammars and for the automatic extraction of such resources. In two companion papers (Frank, 2000; Sadler, van Genabith and Way, 2000) have developed methods for automatically annotating treebank resources with f-structure information. However, to date, these methods have only been applied to treebank fragments of the order of a few hundred trees. In the present paper we present a new method that scales and has been applied to a complete treebank, in our case the WSJ section of Penn-II (Marcus et al, 1994), with more than 1,000,000 words in about 50,000 sentences.

We first give a brief review of Lexical-Functional Grammar. We next review previous work and present three architectures for automatic annotation of treebank resources with f-structure information. We then introduce our new f-structure annotation algorithm and apply it to the Penn-II treebank resource. Finally we conclude and outline further work.

2. Lexical-Functional Grammar

Lexical-Functional Grammar (LFG) is an early member of the family of unification- (more correctly: constraint-) based grammar formalisms (FUG, PATR-II, GPSG, HPSG etc.). It enjoys continued popularity in theoretical and computational linguistics and natural language processing applications and research. At its most basic, an LFG involves two levels of representation: c-structure (constituent structure) and f-structure (functional structure). C-structure represents surface grammatical configurations such as word order and the grouping of linguistic units into larger phrases. The c-structure component of an LFG is represented by a CF-PSG (context-free phrase structure grammar). F-structure represents abstract syntactic functions such as subject, object, predicate etc. in terms of recursive attribute-value structure representations. These abstract syntactic representations abstract away from particulars of surface configuration. The motivation is that while languages differ with respect to surface representation they may still encode the same (or very similar) abstract syntactic functions (or predicate argument structure). To give a simple example, typologically, English is classified as an SVO (subject-verb-object) language while Irish is a verb initial VSO language. Yet a sentence like *John saw Mary* and its Irish translation *Chonaic Seán Máire*, while associated with very different c-structure trees, have structurally isomorphic f-structure representations, as represented in **Figure 1**.

C-structure trees and f-structures are related in terms of projections (indicated by the arrows in the examples in **Figure 1**). These projections are defined in terms of f-structure annotations in c-structure trees (describing f-structures) originating from annotated grammar rules and lexical entries. A sample set of LFG grammar rules with functional annotations (f-descriptions) is provided in **Figure 2**. Optional constituents are indicated by brackets.

3. Previous Work: Automatic Annotation Architectures

It would be desirable to have a treebank annotated with f-structure information as a training resource for probabilistic constraint (unification) grammars and as a resource for extracting such grammars. The large number of CFG rule types in treebanks ($> 19,000$ for Penn-II) makes manual f-structure annotation of grammar rules extracted from complete treebanks prohibitively time consuming and expensive. Recently, in two companion papers (Frank, 2000; Sadler, van Genabith and Way, 2000) a number of researchers have investigated the possibility of automatically annotating treebank resources with f-structure information. As far as we are aware, we can distinguish three different types of automatic f-structure annotation architectures (these have all been developed within an LFG framework and although we refer to these as automatic f-structure an-

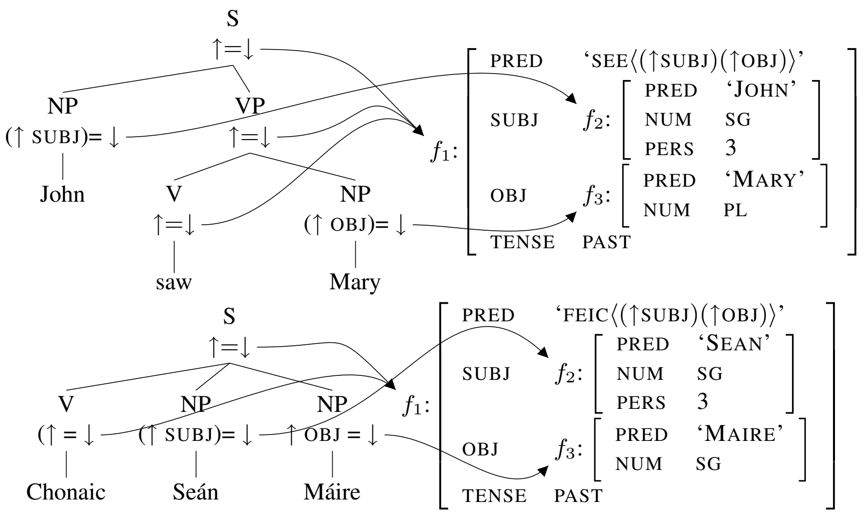

S
↑=↓
NP
(↑ SUBJ)= ↓
VP
↑=↓
John
V
↑=↓
NP
(↑ OBJ)= ↓
saw
Mary
f1:
PRED ‘SEE⟨(↑SUBJ)(↑OBJ)⟩’
SUBJ f2: PRED ‘JOHN’ NUM SG PERS 3
OBJ f3: PRED ‘MARY’ NUM PL
TENSE PAST
S
↑=↓
V
(↑ = ↓
NP
(↑ SUBJ)= ↓
NP
↑ OBJ = ↓
Chonaic
Seán
Máire
f1:
PRED ‘FEIC⟨(↑SUBJ)(↑OBJ)⟩’
SUBJ f2: PRED ‘SEAN’ NUM SG PERS 3
OBJ f3: PRED ‘MAIRE’ NUM SG
TENSE PAST

Figure 1: C- and f-structures for an English and corresponding Irish sentence

$$
\begin{array}{lllll}
S \rightarrow & \begin{array}{c}NP\\ \uparrow SUBJ = \downarrow\end{array} & \begin{array}{c}VP\\ \uparrow = \downarrow\end{array} & \left(\begin{array}{c}ADV\\ \downarrow \in \uparrow ADJN\end{array}\right) & \\[2ex]
NP \rightarrow & \begin{array}{c}Det\\ \uparrow = \downarrow\end{array} & \begin{array}{c}N\\ \uparrow = \downarrow\end{array} & & \\[2ex]
VP \rightarrow & \begin{array}{c}V\\ \uparrow = \downarrow\end{array} & \left(\begin{array}{c}NP\\ \uparrow OBJ = \downarrow\end{array}\right) & \left(\begin{array}{c}VP\\ \uparrow XCOMP = \downarrow\end{array}\right) & \left(\begin{array}{c}S\\ \uparrow COMP = \downarrow\end{array}\right)
\end{array}
$$

Figure 2: Sample LFG grammar rules for a fragment of English

notation architectures they could equally well be used to annotate treebanks with e.g. HPSG feature structure or with Quasi-Logical Form (QLF) (Liakata and Pulman, 2002) annotations):

- regular expression based annotation (Sadler, van Genabith and Way, 2000)
- tree description set based rewriting (Frank, 2000)
- annotation algorithms

More recently, we have learnt about the QLF annotation work by (Liakata and Pulman, 2002). Much like (Frank, 2000), their approach is based on matching configurations in a flat, set based tree description representation.

Below we will briefly describe the first two architectures. The new work presented in this paper is based on an annotation algorithm and discussed at length in Sections 4 and 5 of the paper.

3.1. Regular Repression Based Annotation

(Sadler, van Genabith and Way, 2000) describe a regular expression based automatic f-structure annotation methodology. The basic idea is very simple: first, the CFG rule set is extracted from the treebank (fragment); second, regular expression based annotation principles are defined; third, the principles are automatically applied to the rule set to generate an annotated rule set; fourth, the annotated rules are automatically matched against the original treebank trees and thereby f-structures are generated for these trees. Since the annotation principles factor out linguistic generalisations their number is much smaller than the number of CFG treebank rules. In fact, the regular expression based f-structure annotation principles constitute a principle-based LFG c-structure/f-structure interface. We will explain the method in terms of a simple example. Let us assume that from the treebank trees we extract CFG rules expanding `vp` of the form (amongst others):

```
vp:A > v:B s:C
vp:A > v:B v:C s:D
vp:A > v:B v:C v:D s:E
      ..
vp:A > v:B s:C pp:D
vp:A > v:B v:C s:D pp:E
vp:A > v:B v:C v:D s:E pp:F
      ..
vp:A > advp:B v:C s:D
vp:A > advp:B v:C v:D s:E
vp:A > advp:B v:C v:D v:E s:F
      ..
vp:A > advp:B v:C s:D pp:E
vp:A > advp:B v:C v:D s:E pp:F
vp:A > advp:B v:C v:D v:E s:F pp:G
```

Each CFG category in the rule set has been associated with a logical variable designed to carry f-structure information. In order to annotate these rules we can define a set of regular expression based annotation principles:

```
vp:A > * v:B v:C *
```

```
      @ [B:xcomp=C,B:subj=C:subj]
vp:A > *(~v) v:B *
      @ [A=B]
vp:A > * v:B s:C *
      @ [B:comp=C]
```

The first annotation principle states that if anywhere in a rule RHS expanding a vp category we find a v v sequence the f-structure associated with the second v is the value of an xcomp attribute in the f-structure associated in the first v ('*' is the Kleene star and, if unattached to any other regular expression, signifies any string). It is easy to see how this annotation principle matches many of the extracted example rules, some even twice. The second principle states that the leftmost v in vp rules is the head. The leftmost constraint is expressed by the fact that the rule RHS may consist of an initial string that may not contain a v: *(~v). Each of the annotation principles is partial and underspecified: they underspecify CFG rule RHSs and annotate matching rules partially. The annotation interpreter applies all annotation principles to each CFG rule as often as possible and collects all resulting annotations. It is easy to see that we get, e.g., the following (partial) annotation for:

```
vp:A > advp:B v:C v:D v:E s:F pp:G
      @ [A=C,
         C:xcomp=D,C:subj=D:subj,
         D:xcomp=E,D:subj=E:subj,
         E:comp=F]
```

In their experiments with the publicly available subsection of the AP treebank, (Sadler, van Genabith and Way, 2000) achieve precision and recall results in the low to mid 90 percent region against a manually annotated "gold standard". The method is order independent, partial and robust. To date, however, the method has been applied to only small CFG rule sets (of the order of 500 rules approx.).

3.2. Rewriting of Flat Tree Description Set Representations

In a companion paper, (Frank, 2000) develops an automatic annotation method that in many ways is a generalisation of the regular expression based annotation method. The basic idea is again simple: first, trees in treebanks are translated into a flat set representation format in a tree description language; second, annotation principles are defined in terms of rewriting rules employing a rewriting system originally developed for transfer based machine translation architectures (Kay, 1999). We will illustrate the method with a simple example

```
      s:A
    /    \              dom(A,B), dom(A,C),
np:B    vp:C            dom(C,D), ..
  |       |      =>     pre(B,C),
John     v:D            cat(A,s), cat(C,vp),
          |             cat(D,v), ..
        left


   dom(X,Y), dom(X,Z), pre(Y,Z),
   cat(X,s), cat(Y,np), cat(Z,vp)
             ==>
   subj(X,Y), eq(X,Z)
```

Trees are described in terms of (immediate and general) dominance and precedence relations, labelling functions assigning categories to nodes and so forth. In our example node identifiers A, B, etc. do double duty as f-structure variables. The annotation principle states that if node X dominates both Y and Z and if Y preceeds Z and the respective CFG categories are s, np and vp then Y is the subject of X and Z is the same as (i.e. is the head of) X.

The tree description rewriting method has a number of advantages:

- in contrast to the regular expression based method, annotation principles formulated in the flat tree description method can consider arbitrary tree fragments (and not just only local CFG rule configurations).
- in contrast to the regular expression based method which is order independent, the rewriting technology can be used to formulate both order dependent and order independent systems. Cascaded, order dependent systems can support a more compact and perspicuous statement of annotation principles as certain transformations can be assumed to have already applied earlier on in the cascade.

For a more detailed, joint presentation of the two approaches consult (Frank et al, 2002). Like the regular expression based annotation method, the tree description based set rewriting method has to date only been applied to small treebank fragments of the order of serveral hundred trees.

3.3. Annotation Algorithms

The previous two automatic annotation architectures enforce a clear separation between the statement of annotation principles and the annotation procedure. In the first case the annotation procedure is provided by our regular expression interpreter, in the second by the set rewriting machinery. A clean separation between principles and processing supports maintenance and reuse of annotation principles. There is, however, a third possible automatic annotation architecture and this is an annotation algorithm. In principle, two variants are possible. An annotation algorithm may

- directly (recursively) transduce a treebank tree into an f-structure – such an algorithm would more appropriately be referred to as a tree to f-structure transduction algorithm;
- annotate CFG treebank trees with f-structure annotations from which an f-structure can be computed by a constraint solver.

The first mention of an automatic f-structure annotation algorithm we are aware of is unpublished work by Ron Kaplan (p.c.) who as early as 1996 worked on automatically generating f-structures from the ATIS corpus to generate data for LFG-DOP (Bod and Kaplan, 1998) applications.

Kaplan's approach implements a direct tree to f-structure transduction. The algorithm walks the tree looking for different configurations (e.g. np under s, 2nd np under vp, etc.) and "folds" the tree into the corresponding f-structure. By contrast, our approach develops the second, more indirect tree annotation algorithm paradigm. We have designed and implemented an algorithm that annotates nodes in the Penn-II treebank trees with f-structure constraints. The design and the application of the algorithm is explained below.

4. Automatic Annotation Algorithm Design

In our work on the automatic annotation algorithm we want to achieve the following objectives: we want an annotation method that is robust and scales to the whole of the Penn-II treebank with 19,000 CFG rules for 1,000,000 words with 50,000 sentences approx. The algorithm is implemented as a recursive procedure (in Java) which annotates Penn-II treebank tree nodes with f-structure information. The annotations describe what we call "proto-f-structures". Proto-f-structures

- encode basic predicate-argument-modifier structures;
- may be partial or unconnected (i.e. in some cases a sentence may be associated with two or more unconnected f-structure fragments rather than a single f-structure);
- may not encode some reentrancies, e.g. in the case of wh- and other movement or distribution phenomena (of subjects into VP coordinate structures etc.).

Compared to the regular expression and the set rewriting based annotation methods described above, the new algorithm is somewhat more coarse grained, both with respect to resulting f-structures and with respect to the formulation of the annotation principles.

Even though the method is encoded in the form of an annotation algorithm (i.e. a procedure) we did not want to completely hard code the linguistic basis for the annotation into the procedure. In order to achieve a clean design which supports maintainability and reusability of the annotation algorithm and the linguistic information encoded in it, we decided to design the algorithm in terms of three main components that work in sequence:

$$\begin{tikzcd}
\boxed{\text{L/R Context Annotation Principles}} \arrow[d, Rightarrow] \\
\boxed{\text{Coordinate Annotation Principles}} \arrow[d, Rightarrow] \\
\boxed{\text{Catch-all Annotation Principles}}
\end{tikzcd}$$

Each of the components of the algorithm is presented below.

In addition, at the lexical level, for each Penn-II preterminal category type, we have a lexical macro associating any terminal under the category with the required f-structure information. To give a simple example, a singular common noun nns, such as e.g. *company* is annotated by the lexical macro for nns as $\uparrow$ pred = company, $\uparrow$ num = sg, $\uparrow$ pers = 3rd.

4.1. L/R Context Annotation Principles

The annotation algorithm recursively traverses trees in a top-down fashion. Apart from very few exceptions (e.g. possessive NPs), at each stage of the recursion the algorithm considers local subtrees of depth one (i.e. effectively CFG rules). Annotation is driven by categorial and simple configurational information in a local subtree.

In order to annotate the nodes in the trees, we partition each sequence of daughters in a local subtree (i.e. rule RHS) into three sections: left context, head and right context. The head of a local tree is computed using Collins' Collins (1999) head lexicalised grammar annotation scheme (except for coordinate structures, where we depart from Collins' head scheme). In a preprocessing step we transform the treebank into head lexicalised form. During automatic annotation we can then easily identify the head constituent in a local tree as that constituent which carries the same terminal string as the mother of the local tree. With this we can compute left and right context: given the head constituent, the left context is the prefix of the local daughter sequence while the right context is the suffix. For each local tree we also keep track of the mother category. In addition to the positional (reduced to the simple tripartition into head with left/right context) and categorial information about mother and daughter nodes we also employ an LFG distinction between subcategorisable (subj, obj, obj2, obl, xcomp, comp ...) and non-subcategorisable (adjn, xadjn ...) grammatical functions. Subcategorisable grammatical functions characterise arguments, while non-subcategorisable functions characterise adjuncts (modifiers).

Using this information we construct what we refer to as an "annotation matrix" for each of the rule LHS categories in the Penn-II treebank grammar. The x-axis of the matrix is given by the tripartition into left context, head and right context. The y-axis is defined by the distinction between subcategorisable and non-subcategorisable grammatical functions.

Consider a much simplified example: for rules (local trees) expanding English np's the rightmost nominal (n, nn, nns etc.) on the RHS is (usually) the head. Heads are annotated $\uparrow=\downarrow$. Any det or quant constituent in the left context is annotated $\uparrow$ spec $=\downarrow$. Any adjp in the left context is annotated $\downarrow\in\uparrow$ adjn. Any nominal in the left context (in noun noun sequences) is annotated as a modifier $\downarrow\in\uparrow$ adjn. Any pp in the right context is annotated as $\downarrow\in\uparrow$ adjn. Any relcl in the right context as $\downarrow\in\uparrow$ relmod, any nominal (phrase - usually separated by commas following the head) as an apposition $\downarrow\in\uparrow$ app and so forth. Information such as this is used to populate the np annotation matrix, partially represented in **Table 1**.

In order to minimise mistakes, the annotation matrices are very conservative: subcategorisable grammatical functions are only assigned if there is no doubt (e.g. an np following a preposition in a pp is assigned $\uparrow$ obj $=\downarrow$; a vp following a v in a vp constituent is assigned $\uparrow$ xcomp $=\downarrow$, $\uparrow$ subj $=\uparrow$ xcomp : subj and so forth). If, for any constituent, the argument - modifier status is in doubt, we annotate the constituent as an adjunct: $\downarrow\in\uparrow$ adjn.

Treebanks have an interesting property: for each cate-

np	left context	head	right context
subcat functions	det, quant : ↑ spec =↓	n, nn, nns : ↑=↓	...
non-subcat functions	adjp : ↓∈↑ adjn n, nn, nns : ↓∈↑ adjn ...		relcl : ↓∈↑ relmod pp : ↓∈↑ adjn n, nn, nns : ↓∈↑ app

Table 1: Simplified, partial annotation matrix for np rules

gory, there is a small number of very frequently occurring rules expanding that category, followed by a large number of less frequent rules many of which occur only once or twice in the treebank (Zipf's law).

For each particular category, the corresponding annotation matrix is constructed from the most frequent rules expanding that category. In order to guarantee similar coverage for the annotation matrices for the different rule LHS in the Penn-II treebank, we design each matrix according to an analysis of the most frequent CFG rules expanding that category, such that the token occurrences of those rules cover more than 80% of the token occurrences of all rules expanding that LHS category in the treebank. In order to do this we need to look at the following number of most frequent rule types for each category given in **Table 2**.

Although constructed based on the evidence of the most frequent rule types, the resulting annotation matrices do generalise to as yet unseen rule types in the following two ways:

- during the application of the annotation algorithm, annotation matrices annotate less frequent, unseen rules with constituents matching the left/right context and head specifications. The resulting annotation might be partial (i.e. some constituents in less frequent rule types may be left unannotated).

- in addition to monadic categories, the Penn-II treebank contains versions of these categories associated with functional annotations (-LOC, -TMP etc. indicating locative, temporal, etc. and other functional information). If we include functional annotations in the categories there are approx. 150 distinct LHS categories in the CFG extracted from the Penn-II treebank resource. Our annotation matrices were developed with the most frequent rule types expanding monadic categories only. During application of the annotation algorithm, the annotation matrix for any given monadic category C is also applied to all rules (local trees) expanding C-LOC, C-TMP etc., i.e. instances of the category carrying functional information.

In our work to date we have not yet covered "constituents" marked frag(ment) and x (unknown constituents) in the Penn-II treebank.

Finally, note that L/R context annotation principles are only applied if the local tree (rule RHS) does not contain any instance of a coordinating conjunction cc. Constructions involving coordinating conjunctions are treated separately in the second component of the annotation algorithm.

4.2. Coorordinating Conjunction Annotation Principles

Coordinating constructions come in two forms: like and unlike (UCPs) constituent coordinations. Due to the (often too) flat treebank analyses these present special problems. Because of this, an integrated treatment of coordinate structures with the other annotation principles would have been too complex and messy. For this reason we decided to treat coordinate structures in a separate module. Here we only have space to talk about like constituent coordinations.

The annotation algorithm first attempts to establish the head of a coordinate structure (usually the rightmost coordination) and annotates it accordingly. It then uses a variety of heuristics to find and annotate the various coordinated elements. One of the heuristics employed simply states that if both the immediate left and the immediate right constituents next to the coordination have the same category, then find all such categories in the left context of the rule and annotate these together with the immediate left and right constituents of the coordination as individual elements ↓∈↑ coord in the f-structure set representation of the coordination.

4.3. Catch-All Annotation Principles

The final component of the algorithm utilises functional information provided in the Penn-II treebank annotations. Any constituent, no matter what category, left unannotated by the previous two annotation algorithm components, that carries a Penn-II functional annotation other than SBJ and PRD, is annotated as an adjunct ↓∈↑ adjn.

5. Results and Evaluation

The annotation algorithm is implemented in terms of a Java program. Annotation of the complete WSJ section of the Penn-II treebank takes less than 30 minutes on a Pentium IV PC. Once annotated, for each tree we collect the feature structure annotations and feed them into a simple constraint solver implemented in Prolog.

Our constraint solver can handle equality constraints, disjunction and simple set valued feature constraints. Currently, however, our annotations do not involve disjunctive constraints. This means that for each tree in the treebank we either get a single f-structure, or, in the case of partially annotated trees, a number of unconnected f-structure fragments, or, in case of feature structure clashes, no f-structure.

As pointed out above, in our work to date we have not developed an annotation matrix for frag(mentary) constituents. Furthermore, as it stands, the algorithm completely ignores "movement" (or dislocation and control)

ADJP	ADVP	CONJP	FRAG	LST	NAC	NP	NX	PP	PRN	PRT	QP	RRC
25	3	3	184	4	6	64	14	2	35	2	11	12

S	SBAR	SBARQ	SINV	SQ	UCP	VP	WHADJP	WHADVP	WHNP	WHPP	X	
11	3	20	16	68	78	146	2	2	2	1	37	

Table 2: # of most frequent rule types analysed to construct annotation matrices

phenomena marked in the Penn-II annotations in terms of coindexation (of traces). This means that the f-structures generated in our work to date miss some reentrancies a more fine-grained analysis would show.

Furthermore, because of the limited capabilities of our constraint solver, in our current work we cannot use functional uncertainty constraints (regular expression based constraints over paths in f-structure) to localise unbounded dependencies to model "movement" phenomena. Also, again because of limitations of our constraint solver, we cannot express subsumption constraints in our annotations to, e.g., distribute subjects into coordinate `vp` structures.

To give an illustration of our method, we give the first sentence of the Penn-II treebank and the f-structure generated as an example in **Figure 3**.

Currently we get the following general results with our automatic annotation algorithm summarised in **Table 3**:

# f-structure (fragments)	# sentences	percentage
0	2701	5.576
1	38188	78.836
2	4954	10.227
3	1616	3.336
4	616	1.271
5	197	0.407
6	111	0.229
7	34	0.070
8	12	0.024
9	6	0.012
10	4	0.008
11	1	0.002

Table 3: Automatic annotation results

The Penn-II treebank contains 49167 trees. The results reported in **Table 3** ignore 727 trees containing `frag`(ment) and `x` (unknown) constituents as we did not provide any annotation for them in our work to date. At this early stage of our work, 38188 of the trees are associated with a complete f-structure. For 2701 trees no f-structure is produced (due to feature clashes). 4954 are associated with 2 f-structure fragments, 1616 with 3 fragments and so forth.

5.1. Evaluation

In order to evaluate the results of our automatic annotation we distinguish between "qualitative" and "quantitative" evaluation. Qualitative evaluation involves a "gold-standard", quantitative evaluation doesn't.

5.1.1. Qualitative Evaluation

Currently, we evaluate the output generated by our automatic annotation qualitatively by manually inspecting the f-structures generated. In order to automate the process we are currently working on a set of 100 randomly selected sentences from the Penn-II treebank to manually construct gold-standard annotated trees (and hence f-structures). These can then be processed in a number of ways:

- manually annotated gold-standard trees can be compared with the automatically annotated trees using the labelled bracketing precision and recall measures from `evalb`, a standard software package to evaluate PCFG parses. This presupposes that we treat annotated tree nodes as atoms (i.e. a complex string such as `np:`$\uparrow$ `obj =`$\downarrow$ is treated as an atomic label) and that in cases where nodes receive more than one f-structure annotation the order of these is the same in both the gold-standard and the automatically annotated version.

- gold-standard and automatically generated f-structures can be translated into a flat set of functional descriptions (`pred(A,see), subj(A,B), pred(B,John), obj(A,C), pred(C,Mary)`) and precision and recall can be computed for those.

- f-structures can be transformed (or unfolded) into trees by sorting attributes alphabetically at each level of embedding and by coding reentrancies as indices. After this transformation, gold-standard and automatically generated f-structures can be compared using `evalb`. This presupposes that both the gold-standard and the automatically generated f-structure have identical "terminal" yield.

5.1.2. Quantitative Evaluation

For purely quantitative evaluation (that is evaluation that doesn't necessarily assess the quality of the generated resources) we currently employ two related measures. These measures give an indication how partial our automatic annotation is at the current stage of the project. The first measure is the percentage of RHS constituents in grammar rules that receive an annotation. The table lists the annotation percentage for RHS elements of some of the Penn-II LHS categories. Because of the functional annotations provided in Penn-II the complete list of LHS categories would contain approx. 150 entries. Note that the percentages listed below ignore punctuation markers (which are not annotated):

```
Pierre Vinken, 61 years old, will join the board as a nonexecutive
director Nov. 29.


( S ( NP-SBJ ( NP ( NNP Pierre ) ( NNP Vinken ) ) ( , , ) ADJP ( NP (
 CD 61 ) ( NNS years ) ) ( JJ old ) ) ( , , ) ) ( VP ( MD will ) ( VP
 ( VB join ) ( NP ( DT the ) ( NN board ) ) ( PP-CLR ( IN as ) ( NP (
 DT a ) ( JJ nonexecutive ) ( NN director ) ) ) ( NP-TMP ( NNP Nov. )
 CD 29 ) ) ) ) ( . . ) )


subj : headmod : 1 : num : sing
                     pers : 3
                     pred : Pierre
       num : sing
       pers : 3
       pred : Vinken
       adjunct : 2 : adjunct : 3 : adjunct : 4 : pred : 61
                                   pers : 3
                                   pred : years
                                   num : pl
                     pred : old
xcomp : subj : headmod : 1 : num : sing
                             pers : 3
                             pred : Pierre
               num : sing
               pers : 3
               pred : Vinken
               adjunct : 2 : adjunct : 3 : adjunct : 4 : pred : 61
                                           pers : 3
                                           pred : years
                                           num : pl
                             pred : old
        obj : spec : det : pred : the
              num : sing
              pers : 3
              pred : board
        obl : obj : spec : det : pred : a
                    adjunct : 5 : pred : nonexecutive
                    pred : director
                    num : sing
                    pers : 3
              pred : as
        pred : join
        adjunct : 6 : pred : Nov.
                      num : sing
                      pers : 3
                      adjunct : 7 : pred : 29
pred : will
modal : +
```

Figure 3: F-structure generated for the first sentence in Penn-II

LHS	# RHS elements	# RHS annotated	% annotated
ADJP	1653	1468	88.80
ADJP-ADV	21	21	100.00
ADJP-CLR	27	24	88.88
ADV	607	532	87.64
NP	30793	29145	94.64
PP	1090	905	83.02
S	14912	13144	88.14
SBAR	423	331	78.25
SBARQ	270	212	78.51
SQ	657	601	91.47
VP	40990	35693	87.07

The second, related measure gives the average number of f-structure fragments generated for each treebank tree (the more partial our annotation the more unconnected f-structure fragments are generated for a sentence). For 45739 sentences, the average number of fragments per sentences is currently: 1.26 (note again that the number excludes sentences containing `frag` and `x` constituents).

6. Conclusion and Further Work

In this paper we have presented an automatic f-structure annotation algorithm and applied it to annotate the Penn-II treebank resource with f-structure information. The resulting representations are proto-f-structures showing basic predicate-argument-modifier structure. Currently, 38,188 sentences (78.8% of the 48,440 trees without `frag` and `x` constituents) receive a complete f-structure; 4954 sentences are associated with two f-structure fragments, 1,616 with three fragments. 2,701 sentences are not associated with an f-structure.

In future work we plan to extend and refine our automatic annotation algorithm in a number of ways:

- We are working on reducing the the amount of f-structure fragmentation by providing more complete annotation principles.

- Currently the `pred` values (i.e. the predicates) in the f-structures generated are surface (i.e. inflected) rather than root forms. We are planning to use the output of a two-level morphology to annotate the Penn-II strings with root forms which can then be picked up by our lexical macros and used as `pred` values in the automatic annotations.

- Currently our annotation algorithm ignores the Penn-II encoding of "moved" constituents in topicalisation, wh-constructions, control constructions and the like. These (often non-local) dependencies are marked in the Penn-II tree annotations in terms of indices. In future work we intend to make our annotation algorithm sensitive to such information. There are two (possibly complementary) ways of achieving this: The first is to make the annotation algorithm sensitive to the index scheme provided by the Penn-II annotations either during application of the algorithm or in terms of undoing "movement" in a treebank preprocessing step. The latter route is explored in recent work by (Liakata and Pulman, 2002). The second possibility is to use the LFG machinery of functional uncertainty equations to effectively localise unbounded dependency relations in a functional annotation at a particular node. Functional uncertainty equations allow the statement of regular expression based paths in f-structure. Currently we cannot resolve such paths with our constraint solver.

- We are currently experimenting with probabilistic grammars extracted from the automatically annotated version of the Penn-II treebank. We will be reporting on the results of these experiments elsewhere (Cahill et al, 2002).

- We are planning to exploit the f-structure/QLF/UDRS correspondences established by (van Genabith and Crouch, 1996; van Genabith and Crouch, 1997) to generate semantically annotated versions of the Penn-II treebank.

Acknowledgements

This research was part funded by Enterprise Ireland Basic Research grant SC/2001/186.

7. References

R. Bod and R. Kaplan 1998. A probabilistic corpus-driven model for lexical-functional grammar. In: *Proceedings of Coling/ACL'98*. 145–151.

A. Cahill, M. McCarthy, J. van Genabith and A. Way 2002. Parsing with a PCFG Derived from Penn-II with an Automatic F-Structure Annotation Procedure. In: *The sixth International Conference on Lexical-Functional Grammar*, Athens, Greece, 3 July - 5 July 2002 to appear (2002)

M. Collins 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia.

J. Bresnan 2001. *Lexical-Functional Syntax*. Blackwell, Oxford.

A. Frank. 2000. Automatic F-Structure Annotation of Treebank Trees. In: (eds.) M. Butt and T. H. King, *The fifth International Conference on Lexical-Functional Grammar*, The University of California at Berkeley, 19 July - 20 July 2000, CSLI Publications, Stanford, CA.

A. Frank, L. Sadler, J. van Genabith and A. Way 2002. From Treebank Resources tp LFG F-Structures. In: (ed.) Anne Abeille, *Treebanks: Building and Using Syntactically Annotated Corpora*, Kluwer Academic Publishers, Dordrecht/Boston/London, to appear (2002)

M. Kay 1999. Chart Translation. In *Proceedings of the Machine Translation Summit VII*. "MT in the great Translation Era". 9–14.

R. Kaplan and J. Bresnan 1982. Lexical-functional grammar: a formal system for grammatical representation. In Bresnan, J., editor 1982, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge Mass. 173–281.

M. Liakata and S. Pulman 2002. *From trees to predicate-argument structures*. Unpublished working paper. Centre for Linguistics and Philology, Oxford University.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, M. Ferguson, K. Katz and B. Schasberger 1994. The Penn Treebank: Annotating Predicate Argument Structure. In: *Proceedings of the ARPA Human Language Technology Workshop*.

L. Sadler, J. van Genabith and A. Way. 2000. Automatic F-Structure Annotation from the AP Treebank. In: (eds) M. Butt and T. H. King, *The fifth International Conference on Lexical-Functional Grammar*, The University of California at Berkeley, 19 July - 20 July 2000, CSLI Publications, Stanford, CA.

J. van Genabith and D. Crouch 1996. Direct and Underspecified Interpretations of LFG f-Structures. In: *COLING 96*, Copenhagen, Denmark, Proceedings of the Conference. 262–267.

J. van Genabith and D. Crouch 1997. On Interpreting f-Structures as UDRSs. In: *ACL-EACL-97*, Madrid, Spain, Proceedings of the Conference. 402–409.

Incremental Specialization of an HPSG-Based Annotation Scheme

Kiril Simov, Milen Kouylekov, Alexander Simov

BulTreeBank Project
http://www.BulTreeBank.org
Linguistic Modelling Laboratory, Bulgarian Academy of Sciences
Acad. G. Bonchev St. 25A, 1113 Sofia, Bulgaria
kivs@bgcict.acad.bg, mkouylekov@dir.bg, adis_78@dir.bg

Abstract

The linguistic knowledge represented in contemporary language resource annotations becomes very complex. Its acquiring and management requires an enormous amount of human work. In order to minimize such a human effort we need rigorous methods for representation of such knowledge, methods for supporting the annotation process, methods for exploiting all results from the annotation process, even those that usually disappear after the annotation has been completed. In this paper we present a formal set-up for annotation within HPSG linguistic theory. We present also an algorithm for annotation scheme specialization based on the negative information from the annotation process. The negative information includes the analyses, rejected by the annotator.

1. Introduction

In our project (Simov et. al., 2001a), (Simov et al., 2002) we aim at the creation of syntactically annotated corpus (treebank) based on the HPSG linguistic theory (Head-driven Phrase Structure Grammar — (Pollard and Sag, 1987) and (Pollard and Sag, 1994)). Hence, the elements of the treebank are not trees, but feature graphs. The annotation scheme for the construction of the treebank is based on the appropriate language-specific version of the HPSG sort hierarchy. On one hand, such an annotation scheme is very detailed and flexible with respect to the linguistic knowledge, encoded in it. But, on the other hand, because of the massive overgeneration, it is not considered to be annotator-friendly. Thus, the main problem is: how to keep the consistency of the annotation scheme and at the same time to minimize the human work during the annotation. In our annotation architecture we envisage two sources of linguistic knowledge in order to reduce the possible analyses of the annotated sentences:

- Reliable partial grammars.
- HPSG-based grammar: universal principles, language specific principles and a lexicon.

The actual annotation process includes the following steps:

- Partial parsing step:

  This step comprises several additional steps: (1) Sentence extraction from the text archive; (2) Morphosyntactic tagging; (3) Part-of-speech disambiguation; (4) Partial parsing;

  The result is considered a 100 % accurate partial parsed sentence.

- HPSG step:

  The result from the previous step is encoded into an HPSG compatible representation with respect to the sort hierarchy. It is sent to an HPSG grammar tool, which takes the partial sentence analysis as input and evaluates all the attachment possibilities for it. The output is encoded as feature graphs.

- Annotation step:

  The feature graphs from the previous step are further processed as follows : (1) their intersection is calculated; (2) on the base of the differences, a set of constraints over the intersection is calculated as well; (3) during the actual annotation step, the annotator tries to extend the intersection to full analysis, adding new information to it. The constraints determine the possible extensions and also propagate the information, added by the annotator, in order to minimize the incoming choices.

This architecture is being currently implemented by establishing an interface between two systems: CLaRK system for XML based corpora development (Simov et. al., 2001b) and TRALE system for HPSG grammar development (TRALE is a descendant of (Götz and Meurers, 1997)). The project will result in an HPSG corpus based on feature graphs and reliable grammars. One of the intended applications of these language resources consists of their exploration for improving the accuracy of the implemented HPSG grammar.

The work, reported in this paper, is a step towards establishing an incremental mechanism, which uses already annotated sentences for further specializing of the HPSG grammar and for reducing the number of the possible HPSG analyses. In fact, we consider the rejected analyses as negative information about the language and therefore the grammar has to be appropriately tuned in order to rule out such analyses.

The structure of the paper is as follows: in the next section we define formally what a corpus is with respect to a grammar formalism and apply this definition to the definition of an HPSG corpus. In Sect. 3. we present a logical formalism for HPSG, define a normal form for grammars in the logical formalism and on the basis of this normal form we define feature graphs that constitute a good representation for both — HPSG grammars and HPSG corpora. Sect. 4. presents the algorithm for specialization of an

HPSG grammar on the basis of accepted and rejected by the annotator analyses produced by the grammar. Then Sect. 5. demonstrates an example of such specialization. The last section outlines the conclusions and outlook.

2. HPSG Corpus

In our work we accept that the corpus is complete with respect to the analyses of the sentences in it. This means that each sentence is presented with all its acceptable syntactic structures. Thus a good grammar will not overgenerate, i.e. it will not assign more analyses to the sentences than the analyses, which already exist in the corpus. Before we define what an HPSG corpus is like, let us start with a definition of a grammar-formalism-based corpus in general. Such an ideal corpus has to ensure the above assumption.

Definition 1 (Grammar Formalism Corpus) *A corpus C in a given grammatical formalism G is a sequence of analyzed sentences where each analyzed sentence is a member of the set of structures defined as a* **strong generative capacity** *(*SGC*) of a grammar* Γ *in this grammatical formalism:*

$\forall S.S \in C \rightarrow S \in \mathtt{SGC}(\Gamma)$,

where Γ *is a grammar in the formalism G, and if* $\sigma(S)$ *is the phonological string of S and* $\Gamma(\sigma(S))$ *is the set of all analyses assigned by the grammar* Γ *to the phonological string* $\sigma(S)$*, then*

$\forall S'.S' \in \Gamma(\sigma(S)) \rightarrow S' \in C.$

The grammar Γ is unknown, but implicitly represented in the corpus C. We could state that if such a grammar does not exist, then we consider the corpus inconsistent or uncomplete.

In order to define a corpus in HPSG with respect to this definition, we have to define a representation of HPSG analysis over the sentences. This analysis must correspond to a definition of strong generative capacity in HPSG. Fortunately, there exist such definitions - (King 1999) and (Pollard 1999). We adopt them for our purposes. Thus in our work we choose:

- A logical formalism for HPSG — King's Logic (SRL) (King 1989);
- A definition of strong generative capacity in HPSG as a set of feature structures closely related to the special interpretation in SRL (exhaustive models) along the lines of (King 1999) and (Pollard 1999).
- A definition of corpus in HPSG as a sequence of sentences that are members of $\mathtt{SGC}(\Gamma)$ for some grammar Γ in SRL.

It is well-known that an HPSG grammar in SRL formally comprises two parts: a signature and a theory. The signature defines the ontology of the linguistic objects in the language and the theory constraints the shape of the linguistic objects. Usually the descriptions in the theory part are presented as implications. In order to demonstrate in a better way the connection between the HPSG grammar in SRL and the HPSG corpus, we offer a common representation of the grammar and the corpus.

We define a normal form for HPSG grammars which ideologically is very close to the feature structures defining the strong generative capacity in HPSG as it has proposed in the work of (King 1999) and (Pollard 1999). We define both the corpus and the grammar in terms of clauses (considered as graphs) in a special kind of matrices in SRL. The construction of new sentence analyses can be done using the inference mechanisms of SRL. Another possibility is such a procedure to be defined directly using the representations in the normal form. In order to distinguish the elements in our normal form from the numerous of kinds of feature structures we call the elements in the normal form *feature graphs*. One important characteristic about our feature graphs is that they are viewed as descriptions in SRL, i.e. as syntactic entities.

In other works (Simov, 2001) and (Simov, 2002) we showed how from a corpus, consisting of feature graphs, a corpus grammar could be extracted along the lines of Rens Bod's ideas on Data-Oriented Parsing Model (Bod, 1998). Also, in (Simov, 2002) we showed how one could use the positive information in the corpus in order to refine an existing HPSG grammar. In this paper we discuss and illustrate the usage of the negative information compiled as a by-product during the annotation of the corpus.

3. Logical Formalism for HPSG

In this section we present a logical formalism for HPSG. Then a normal form (exclusive matrices) for a finite theory in this formalism is defined and then we show how it can be represented as a set of feature graphs. These graphs are considered a representation of grammars and corpora in HPSG.

3.1. King's Logic — SRL

This section presents the basic notions of Speciate Reentrancy Logic (SRL) (King 1989).

$\Sigma = \langle \mathcal{S}, \mathcal{F}, \mathcal{A} \rangle$ is a finite **SRL signature** iff $\mathcal{S}$ is a finite set of *species*, $\mathcal{F}$ is a set of *features*, and $\mathcal{A} : \mathcal{S} \times \mathcal{F} \rightarrow Pow(\mathcal{S})$ is an *appropriateness function*. $\mathcal{I} = \langle \mathcal{U}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{F}_\mathcal{I} \rangle$ is a **SRL interpretation** of the signature Σ (or Σ-interpretation) iff

$\mathcal{U}_\mathcal{I}$ is a non-empty set of objects,
$\mathcal{S}_\mathcal{I}$ is a total function from $\mathcal{U}_\mathcal{I}$ to $\mathcal{S}$,
called **species assignment function**,
$\mathcal{F}_\mathcal{I}$ is a total function from $\mathcal{F}$ to the set of partial function from $\mathcal{U}_\mathcal{I}$ to $\mathcal{U}_\mathcal{I}$ such that
for each $\phi \in \mathcal{F}$ and each $\upsilon \in \mathcal{U}_\mathcal{I}$, if $\mathcal{F}_\mathcal{I}(\phi)(\upsilon)\downarrow$[1]
then $\mathcal{S}_\mathcal{I}(\mathcal{F}_\mathcal{I}(\phi)(\upsilon)) \in \mathcal{A}(\mathcal{S}_\mathcal{I}(\upsilon), \phi)$, and
for each $\phi \in \mathcal{F}$ and each $\upsilon \in \mathcal{U}_\mathcal{I}$, if $\mathcal{A}(\mathcal{S}_\mathcal{I}(\upsilon), \phi)$ is not empty then $\mathcal{F}_\mathcal{I}(\phi)(\upsilon)\downarrow$,
$\mathcal{F}_\mathcal{I}$ is called **feature interpretation function**.

τ is a term iff τ is a member of the smallest set $\mathcal{TM}$ such that (1) $: \in \mathcal{TM}$, and (2) for each $\phi \in \mathcal{F}$ and each $\tau \in \mathcal{TM}$, $\tau\phi \in \mathcal{TM}$. For each Σ-interpretation $\mathcal{I}$, $\mathcal{P}_\mathcal{I}$ is a **term interpretation function** over $\mathcal{I}$ iff (1) $\mathcal{P}_\mathcal{I}(:)$ is the identity function from $\mathcal{U}_\mathcal{I}$ to $\mathcal{U}_\mathcal{I}$, and (2) for each $\phi \in \mathcal{F}$ and each $\tau \in \mathcal{TM}$, $\mathcal{P}_\mathcal{I}(\tau\phi)$ is the composition of the partial functions $\mathcal{P}_\mathcal{I}(\tau)$ and $\mathcal{F}_\mathcal{I}(\phi)$ if they are defined.

[1] $f(o)\downarrow$ means the function f is defined for the argument o.

δ is a **description** iff δ is a member of the smallest set $\mathcal{D}$ such that (1) for each $\sigma \in \mathcal{S}$ and for each $\tau \in \mathcal{TM}$, $\tau \sim \sigma \in \mathcal{D}$, (2) for each $\tau_1 \in \mathcal{TM}$ and $\tau_2 \in \mathcal{TM}$, $\tau_1 \approx \tau_2 \in \mathcal{D}$ and $\tau_1 \not\approx \tau_2 \in \mathcal{D}$, (3) for each $\delta \in \mathcal{D}$, $\neg\delta \in \mathcal{D}$, (4) for each $\delta_1 \in \mathcal{D}$ and $\delta_2 \in \mathcal{D}$, $[\delta_1 \wedge \delta_2] \in \mathcal{D}$, $[\delta_1 \vee \delta_2] \in \mathcal{D}$, and $[\delta_1 \rightarrow \delta_2] \in \mathcal{D}$. **Literals** are descriptions of the form $\tau \sim \sigma$, $\tau_1 \approx \tau_2$, $\tau_1 \not\approx \tau_2$ or their negation. For each Σ-interpretation $\mathcal{I}$, $\mathcal{D}_\mathcal{I}$ is a **description denotation function** over $\mathcal{I}$ iff $\mathcal{D}_\mathcal{I}$ is a total function from $\mathcal{D}$ to the powerset of $\mathcal{U}_\mathcal{I}$, such that

$$\begin{aligned}
&\mathcal{D}_\mathcal{I}(\tau \sim \sigma) = \{\upsilon \in \mathcal{U}_\mathcal{I} \mid \mathcal{P}_\mathcal{I}(\tau)(\upsilon)\downarrow, \\
&\qquad \mathcal{S}_\mathcal{I}(\mathcal{P}_\mathcal{I}(\tau)(\upsilon)) = \sigma\}, \\
&\mathcal{D}_\mathcal{I}(\tau_1 \approx \tau_2) = \{\upsilon \in \mathcal{U}_\mathcal{I} \mid \mathcal{P}_\mathcal{I}(\tau_1)(\upsilon)\downarrow, \mathcal{P}_\mathcal{I}(\tau_2)(\upsilon)\downarrow, \\
&\qquad \text{and } \mathcal{P}_\mathcal{I}(\tau_1)(\upsilon) = \mathcal{P}_\mathcal{I}(\tau_2)(\upsilon)\}, \\
&\mathcal{D}_\mathcal{I}(\tau_1 \not\approx \tau_2) = \{\upsilon \in \mathcal{U}_\mathcal{I} \mid \mathcal{P}_\mathcal{I}(\tau_1)(\upsilon)\downarrow, \mathcal{P}_\mathcal{I}(\tau_2)(\upsilon)\downarrow, \\
&\qquad \text{and } \mathcal{P}_\mathcal{I}(\tau_1)(\upsilon) \neq \mathcal{P}_\mathcal{I}(\tau_2)(\upsilon)\}, \\
&\mathcal{D}_\mathcal{I}(\neg\delta) = \mathcal{U}_\mathcal{I} \setminus \mathcal{D}_\mathcal{I}(\delta), \\
&\mathcal{D}_\mathcal{I}([\delta_1 \wedge \delta_2]) = \mathcal{D}_\mathcal{I}(\delta_1) \cap \mathcal{D}_\mathcal{I}(\delta_2), \\
&\mathcal{D}_\mathcal{I}([\delta_1 \vee \delta_2]) = \mathcal{D}_\mathcal{I}(\delta_1) \cup \mathcal{D}_\mathcal{I}(\delta_2), \text{ and} \\
&\mathcal{D}_\mathcal{I}([\delta_1 \rightarrow \delta_2]) = (\mathcal{U}_\mathcal{I} \setminus \mathcal{D}_\mathcal{I}(\delta_1)) \cup \mathcal{D}_\mathcal{I}(\delta_2).
\end{aligned}$$

Each subset $\theta \subseteq \mathcal{D}$ is an **SRL theory**. For each Σ-interpretation $\mathcal{I}$, $\mathcal{T}_\mathcal{I}$ is a **theory denotation function** over $\mathcal{I}$ iff $\mathcal{T}_\mathcal{I}$ is a total function from the powerset of $\mathcal{D}$ to the powerset of $\mathcal{U}_\mathcal{I}$ such that for each $\theta \subseteq \mathcal{D}$, $\mathcal{T}_\mathcal{I}(\theta) = \cap\{\mathcal{D}_\mathcal{I}(\delta) | \delta \in \theta\}$. $\mathcal{T}_\mathcal{I}(\emptyset) = \mathcal{U}_\mathcal{I}$. A theory θ is **satisfiable** iff for some interpretation $\mathcal{I}$, $\mathcal{T}_\mathcal{I}(\theta) \neq \emptyset$. A theory θ is **modelable** iff for some interpretation $\mathcal{I}$, $\mathcal{T}_\mathcal{I}(\theta) = \mathcal{U}_\mathcal{I}$, $\mathcal{I}$ is called a **model** of θ. The interpretation $\mathcal{I}$ **exhaustively models** θ iff

$\mathcal{I}$ is a model of θ, and
for each $\theta' \subseteq \mathcal{D}$,
if for some model $\mathcal{I}'$ of θ,
$\mathcal{T}_{\mathcal{I}'}(\theta') \neq \emptyset$,
then $\mathcal{T}_\mathcal{I}(\theta') \neq \emptyset$.

An HPSG grammar $\Gamma = \langle \Sigma, \theta \rangle$ in SRL consists of: (1) a signature Σ which gives the ontology of entities that exist in the universe and the appropriateness conditions on them, and (2) a theory θ which gives the restrictions upon these entities.

3.2. Exclusive Matrices

Following (King and Simov, 1998) in this section we define a normal form for finite theories in SRL — called **exclusive matrix**. This normal form possesses some desirable properties for representation of grammars and corpora in HPSG.

First, we define some technical notions. A **clause** is a finite set of literals interpreted *conjunctively*. A **matrix** is a finite set of clauses interpreted *disjunctively*.

A matrix μ is an **exclusive matrix** iff for each clause $\alpha \in \mu$,

(E0) if $\lambda \in \alpha$ then λ is a positive literal,
(E1) $:\approx: \in \alpha$,
(E2) if $\tau_1 \approx \tau_2 \in \alpha$ then $\tau_2 \approx \tau_1 \in \alpha$,
(E3) if $\tau_1 \approx \tau_2 \in \alpha$ and $\tau_2 \approx \tau_3 \in \alpha$ then $\tau_1 \approx \tau_3 \in \alpha$,
(E4) if $\tau\phi \approx \tau\phi \in \alpha$ then $\tau \approx \tau \in \alpha$,
(E5) if $\tau_1 \approx \tau_2 \in \alpha$, $\tau_1\phi \approx \tau_1\phi \in \alpha$ and $\tau_2\phi \approx \tau_2\phi \in \alpha$ then
$\tau_1\phi \approx \tau_2\phi \in \alpha$,
(E6) if $\tau \approx \tau \in \alpha$ then for some $\sigma \in \mathcal{S}$, $\tau \sim \sigma \in \alpha$,
(E7) if for some $\sigma \in \mathcal{S}$, $\tau \sim \sigma \in \alpha$ then $\tau \approx \tau \in \alpha$,
(E8) if $\tau_1 \approx \tau_2 \in \alpha$, $\tau_1 \sim \sigma_1 \in \alpha$ and $\tau_2 \sim \sigma_2 \in \alpha$ then
$\sigma_1 = \sigma_2$,
(E9) if $\tau \sim \sigma_1 \in \alpha$ and $\tau\phi \sim \sigma_2 \in \alpha$ then $\sigma_2 \in \mathcal{A}(\sigma_1, \phi)$,
(E10) if $\tau \sim \sigma \in \alpha$, $\tau\phi \in \mathit{Term}(\mu)$ and $\mathcal{A}(\sigma, \phi) \neq \emptyset$ then
$\tau\phi \approx \tau\phi \in \alpha$,
(E11) if $\tau_1 \not\approx \tau_2 \in \alpha$ then $\tau_1 \approx \tau_1 \in \alpha$ and $\tau_2 \approx \tau_2 \in \alpha$,
(E12) if $\tau_1 \approx \tau_1 \in \alpha$ and $\tau_2 \approx \tau_2 \in \alpha$ then
$\tau_1 \approx \tau_2 \in \alpha$ or $\tau_1 \not\approx \tau_2 \in \alpha$, and
(E13) $\tau_1 \approx \tau_2 \notin \alpha$ or $\tau_1 \not\approx \tau_2 \notin \alpha$,

where $\{\sigma, \sigma_1, \sigma_2\} \subseteq \mathcal{S}$, $\phi \in \mathcal{F}$, and $\{\tau, \tau_1, \tau_2, \tau_3\} \subseteq \mathcal{TM}$, and `Term` is a function from the powerset of the sets of literals to the powerset of $\mathcal{TM}$ such that

$$\begin{aligned}
\mathtt{Term}(\alpha) = &\{\tau \mid (\neg)\tau\phi \approx \tau' \in \alpha, \tau \in \mathcal{TM}, \phi \in \mathcal{F}^*\} \cup \\
&\{\tau \mid (\neg)\tau' \approx \tau\phi \in \alpha, \tau \in \mathcal{TM}, \phi \in \mathcal{F}^*\} \cup \\
&\{\tau \mid (\neg)\tau\phi \not\approx \tau' \in \alpha, \tau \in \mathcal{TM}, \phi \in \mathcal{F}^*\} \cup \\
&\{\tau \mid (\neg)\tau' \not\approx \tau\phi \in \alpha, \tau \in \mathcal{TM}, \phi \in \mathcal{F}^*\} \cup \\
&\{\tau \mid (\neg)\tau\phi \sim \sigma \in \alpha, \tau \in \mathcal{TM}, \phi \in \mathcal{F}^*\}.
\end{aligned}$$

There are two important properties of an exclusive matrix $\mu = \{\alpha_1, \ldots, \alpha_n\}$: (1) each clause α in μ is satisfiable (for some interpretation $\mathcal{I}$, $\mathcal{T}_\mathcal{I}(\alpha) \neq \emptyset$), and (2) each two clauses α_1, α_2 in μ have disjoint denotations (for each interpretation $\mathcal{I}$, $\mathcal{T}_\mathcal{I}(\alpha_1) \cap \mathcal{T}_\mathcal{I}(\alpha_1) = \emptyset$). Also in (King and Simov, 1998) it is shown that each finite theory with respect to a finite signature can be converted into an exclusive matrix which is semantically equivalent to the theory. Relying on the definition of model (where each object in the domain is described by the theory) and the property that each two clauses in an exclusive matrix have disjoint denotation, one can easy prove the following proposition.

Proposition 2 *Let θ be a finite SRL theory with respect to a finite signature, μ be the corresponding exclusive matrix and $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{F} \rangle$ be a model of θ. For each object $\upsilon \in \mathcal{U}$ there exists a unique clause $\alpha \in \mu$ such that $\upsilon \in \mathcal{T}(\alpha)$.*

3.3. Feature Graphs

As it was mentioned above, an HPSG corpus will comprise a set of feature structures representing the HPSG analyses of the sentences. We interpret these feature structures as descriptions in SRL (clauses in an exclusive matrix).

Let $\Sigma = \langle \mathcal{S}, \mathcal{F}, \mathcal{A} \rangle$ be a finite signature. A directed, connected and rooted graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$ such that

$\mathcal{N}$ is a set of **nodes**,
$\mathcal{V} : \mathcal{N} \times \mathcal{F} \rightarrow \mathcal{N}$ is a partial **arc function**,
ρ is a **root node**,
$\mathcal{S} : \mathcal{N} \rightarrow \mathcal{S}$ is a total **species assignment function**,
such that
for each $\nu_1, \nu_2 \in \mathcal{N}$ and each $\phi \in \mathcal{F}$
if $\mathcal{V}\langle \nu_1, \phi \rangle\downarrow$ and $\mathcal{V}\langle \nu_1, \phi \rangle = \nu_2$,
then $\mathcal{S}\langle \nu_2 \rangle \in \mathcal{A}\langle \mathcal{S}\langle \nu_1 \rangle, \phi \rangle$,

is a **feature graph** wrt Σ.

A feature graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$ such that for each node $\nu \in \mathcal{N}$ and each feature $\phi \in \mathcal{F}$ if $\mathcal{A}\langle \mathcal{S}\langle \nu \rangle, \phi \rangle\downarrow$ then $\mathcal{V}\langle \nu, \phi \rangle\downarrow$ is called a **complete feature graph** (or complete graph).

According to our definition feature graphs are a kind of feature structures which are treated syntactically rather than semantically. We use complete feature graphs for representing the analyses of the sentences in the corpus.

We say that the feature graph $\mathcal{G}$ is **finite** if and only if the set of nodes is finite.

For each graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$ and node ν in $\mathcal{N}$ with $\mathcal{G}\,|_\nu = \langle \mathcal{N}_\nu, \mathcal{V}\,|_{\mathcal{N}_\nu}, \rho_\nu, \mathcal{S}\,|_{\mathcal{N}_\nu} \rangle$ we denote the **subgraph** of $\mathcal{G}$ starting on node ν.

Let $\mathcal{G}_1 = \langle \mathcal{N}_1, \mathcal{V}_1, \rho_1, \mathcal{S}_1 \rangle$ and $\mathcal{G}_2 = \langle \mathcal{N}_2, \mathcal{V}_2, \rho_2, \mathcal{S}_2 \rangle$ be two graphs. We say that graph $\mathcal{G}_1$ **subsumes** graph $\mathcal{G}_2$ ($\mathcal{G}_2 \sqsubseteq \mathcal{G}_1$) iff there is an *isomorphism* $\gamma : \mathcal{N}_1 \rightarrow \mathcal{N}_2'$, $\mathcal{N}_2' \subseteq \mathcal{N}_2$, such that

$\gamma(\rho_1) = \rho_2$,

for each $\nu, \nu' \in \mathcal{N}_1$ and each feature ϕ,

$\mathcal{V}_1\langle \nu, \phi \rangle = \nu'$ iff $\mathcal{V}_2\langle \gamma(\nu), \phi \rangle = \gamma(\nu')$, and

for each $\nu \in \mathcal{N}_1$, $\mathcal{S}_1\langle \nu \rangle = \mathcal{S}_2\langle \gamma(\nu) \rangle$.

The intuition behind the definition of subsumption by isomorphism is that each graph describes "exactly" a chunk in some SRL interpretation in such a way that every two distinct nodes are always mapped to distinct objects in the interpretation.

For each two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ if $\mathcal{G}_2 \sqsubseteq \mathcal{G}_1$ and $\mathcal{G}_1 \sqsubseteq \mathcal{G}_2$ we say that $\mathcal{G}_1$ and $\mathcal{G}_2$ are **equivalent.** For convenience, in the following text we consider each two equivalent graphs equal.

For a finite feature graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$, we define a translation to a clause. Let

$\mathit{Term}(\mathcal{G}) = \{:\} \cup \{\tau \mid \tau \doteq :\phi_1 \ldots \phi_n, n \leq \|\mathcal{N}\|, \mathcal{V}\langle \rho, \tau \rangle \downarrow\}$[2]

be a set of terms. We define a clause $\alpha_\mathcal{G}$:

$\alpha_\mathcal{G} = \{\tau \sim \sigma \mid \tau \in \mathit{Term}(\mathcal{G}), \mathcal{V}\langle \rho, \tau \rangle \downarrow, \mathcal{S}\langle \mathcal{V}\langle \rho, \tau \rangle \rangle = \sigma\} \cup$
$\{\tau_1 \approx \tau_2 \mid \tau_1 \in \mathit{Term}(\mathcal{G}), \tau_2 \in \mathit{Term}(\mathcal{G}),$
$\mathcal{V}\langle \rho, \tau_1 \rangle \downarrow, \mathcal{V}\langle \rho, \tau_2 \rangle \downarrow$, and $\mathcal{V}\langle \rho, \tau_1 \rangle = \mathcal{V}\langle \rho, \tau_2 \rangle\} \cup$
$\{\tau_1 \not\approx \tau_2 \mid \tau_1 \in \mathit{Term}(\mathcal{G}), \tau_2 \in \mathit{Term}(\mathcal{G}),$
$\mathcal{V}\langle \rho, \tau_1 \rangle \downarrow, \mathcal{V}\langle \rho, \tau_2 \rangle \downarrow$, and $\mathcal{V}\langle \rho, \tau_1 \rangle \neq \mathcal{V}\langle \rho, \tau_2 \rangle\}$.

We interpret a finite feature graph via the interpretation of the corresponding clauses

$\mathcal{R}_\mathcal{I}(\mathcal{G}) = \mathcal{T}_\mathcal{I}(\alpha_\mathcal{G})$.

Let $\mathcal{G}$ be an infinite feature graph. Then we interpret it as the intersection of the interpretations of all finite feature graphs that subsume it:

$\mathcal{R}_\mathcal{I}(\mathcal{G}) = \cap_{\mathcal{G} \sqsubseteq \mathcal{G}', \mathcal{G}' < \omega} \mathcal{R}_\mathcal{I}(\alpha_{\mathcal{G}'})$.

The clauses in an exclusive matrix μ can be represented as feature graphs. Let μ be an exclusive matrix and $\alpha \in \mu$, then

$\mathcal{G}_\alpha = \langle \mathcal{N}_\alpha, \mathcal{V}_\alpha, \rho_\alpha, \mathcal{S}_\alpha \rangle$ is a feature graph such that

$\mathcal{N}_\alpha = \{|\tau|_\alpha \mid \tau \approx \tau \in \alpha\}$ is a set of nodes,

$\mathcal{V}_\alpha : \mathcal{N}_\alpha \times \mathcal{F} \rightarrow \mathcal{N}_\alpha$ is a partial **arc function**, such that

$\mathcal{V}_\alpha\langle |\tau_1|_\alpha, \phi \rangle \downarrow$ and $\mathcal{V}_\alpha\langle |\tau_1|_\alpha, \phi \rangle = |\tau_2|_\alpha$ iff

$\tau_1 \approx \tau_1 \in \alpha, \tau_2 \approx \tau_2 \in \alpha, \phi \in \mathcal{F}$, and $\tau_1\phi \approx \tau_2 \in \alpha$,

ρ_α is the root node $|:|_\alpha$, and

$\mathcal{S}_\alpha : \mathcal{N}_\alpha \rightarrow \mathcal{S}$ is a **species assignment function**, such that

$\mathcal{S}_\alpha\langle |\tau|_\alpha \rangle = \sigma$ iff $\tau \sim \sigma \in \alpha$.

Proposition 3 *Let μ be an exclusive matrix and $\alpha \in \mu$. Then the graph $\mathcal{G}_\alpha$ is semantically equivalent to α.*

3.4. Inference with Feature Graphs

In this paper we do not present a concrete inference mechanism exploiting feature graphs. As it was mentioned above, one can use the general inference mechanisms of SRL in order to construct sentence analyses. However, a much better solution is to employ an inference mechanism, which uses directly the graph representation of a theory. Such an inference mechanism can be defined along the lines of *Breadth-First Parallel Resolution* in (Carpener 1992) despite the difference in the treatment of the feature structure in (Carpener 1992) (Note that (Carpener 1992) treats feature structures as semantic entities, but we consider our feature graphs syntactic elements.). One has to keep in mind that finding models in SRL is undecidable (see (King, Simov and Aldag 1999)) and some restrictions in terms of time or memory will be necessary in order to use Breadth-First Parallel Resolution-like algorithm. A presentation of such an algorithm is beyond the scope of this paper.

3.5. Graph Representation of an SRL Theory

Each finite SRL theory can be represented as a set of feature graphs. In order to make this graph transformation of a theory completely independent from the SRL particulars, we also need to incorporate within the graphs the information from the signature that is not present in the theory yet. For each species the signature encodes the defined features as well as the species of their possible values. We explicate this information in the signature by constructing a special theory:

$$\theta_\Sigma = \{ \bigvee_{\sigma \in \mathcal{S}} [\bigwedge_{\mathcal{A}(\sigma,\phi) \neq \emptyset, \phi \in \mathcal{F}} [:\phi \approx :\phi]]\}.$$

Then for each theory θ we form the theory $\theta^e = \theta \cup \theta_\Sigma$ which is semantically equivalent to the original theory (because we add only the information from the signature which is always taken into account, when a theory is interpreted). We convert the theory θ^e into an exclusive matrix which in turn is converted into a set of graphs $\mathcal{GR}$ called **graph representation** of θ.

The graph representation of a theory inherits from the exclusive matrixes their properties: (1) each graph $\mathcal{G}$ in $\mathcal{GR}$ is satisfiable (for some interpretation $\mathcal{I}$, $\mathcal{R}_\mathcal{I}(\mathcal{G}) \neq \emptyset$), and (2) each two graphs $\mathcal{G}_1$, $\mathcal{G}_2$ in $\mathcal{GR}$ have disjoint denotations (for each interpretation $\mathcal{I}$, $\mathcal{R}_\mathcal{I}(\mathcal{G}_1) \cap \mathcal{R}_\mathcal{I}(\mathcal{G}_2) = \emptyset$). We can reformulate here also the Prop. 2.

Proposition 4 *Let θ be a finite SRL theory with respect to a finite signature, μ be the corresponding exclusive matrix, $\mathcal{GR}$ be the graph representation of θ and $\mathcal{I} = \langle \mathcal{U}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{F}_\mathcal{I} \rangle$ be a model of θ. For each object $\upsilon \in \mathcal{U}$ there exists a unique graph $\mathcal{G} \in \mathcal{GR}$ such that $\upsilon \in \mathcal{R}(\mathcal{G})$.*

There exists also a correspondence between complete graphs with respect to a finite signature and the objects in an interpretation of the signature.

Definition 5 (Object Graph) *Let $\Sigma = \langle \mathcal{S}, \mathcal{F}, \mathcal{A} \rangle$ be a finite signature, $\mathcal{I} = \langle \mathcal{U}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{F}_\mathcal{I} \rangle$ be an interpretation of Σ and υ be an object in $\mathcal{U}$, then the graph $\mathcal{G}_\upsilon = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$, where*

$\mathcal{N} = \{\upsilon' \in \mathcal{U} \mid \exists \tau \in \mathcal{TM}$ *and* $\mathcal{P}(\tau)(\upsilon) = \upsilon'\}$

$\mathcal{V} : \mathcal{N} \times \mathcal{F} \rightarrow \mathcal{N}$ *is a partial* **arc function**, *such that*

$\mathcal{V}\langle \upsilon_1, \phi \rangle \downarrow$ *and* $\mathcal{V}\langle \upsilon_1, \phi \rangle = \upsilon_2$ *iff*

$\upsilon_1 \in \mathcal{N}, \upsilon_2 \in \mathcal{N}, \phi \in \mathcal{F}$, *and* $\mathcal{F}_\mathcal{I}(\phi)(\upsilon_1) = \upsilon_2$,

$\rho = \upsilon$ *is the root node, and*

$\mathcal{S} : \mathcal{N} \rightarrow \mathcal{S}$ *is a* **species assignment function**, *such that*

$\mathcal{S}\langle \upsilon' \rangle = \mathcal{S}_\mathcal{I}\langle \upsilon' \rangle$,

is called **object graph**.

[2] $\|X\|$ is the cardinality of the set X

It is trivial to check that each object graph is a complete feature graph. Also, one easy can see the connection between the graphs in the graph representation of a theory and object graphs of objects in a model of the theory.

Proposition 6 *Let θ be a finite SRL theory with respect to a finite signature, $\mathcal{GR}$ be the graph representation of θ, $\mathcal{I} = \langle \mathcal{U}_\mathcal{I}, \mathcal{S}_\mathcal{I}, \mathcal{F}_\mathcal{I} \rangle$ be a model of θ, υ be an object in $\mathcal{U}_\mathcal{I}$, and $\mathcal{G}_\upsilon = \langle \mathcal{N}, \mathcal{V}, \rho, \mathcal{S} \rangle$ be its object graph. For each node $\nu \in \mathcal{N}$, there exists a graph $\mathcal{G}_i \in \mathcal{GR}$, such that $\mathcal{G}\,|_\nu \sqsubseteq \mathcal{G}_i$.*

This can be proved by using the definition of a model of a theory, the Prop. 4 and the definition of a subgraph started at a node.

3.6. Outcomes: Feature Graphs for HPSG Grammar and Corpus

Thus we can sum up that feature graphs can be used for both:

- Representation of an HPSG corpus. Each sentence in the corpus is represented as a complete feature graph. One can easily establish a correspondence between the objects in an exhaustive model of (King 1999) and complete feature graphs or a correspondence between the elements of strong generative capacity of (Pollard 1999) and complete feature graphs. Thus complete feature graphs are a good representation for an HPSG corpus;

- Representation of an HPSG grammar as a set of feature graphs. The construction of a graph representation of a finite theory demonstrates that using feature graphs as grammar representation does not impose any restrictions over the class of possible finite grammars in SRL. Therefore we can use feature graphs as a representation of the grammar used during the construction of an HPSG corpus, as described above.

Additionally, we can establish a formal connection between a grammar and a corpus using the properties of feature graphs.

Definition 7 (Corpus Grammar) *Let C be an HPSG corpus and Γ be an HPSG grammar. We say that Γ is a* **grammar of the corpus** *C if and only if for each graph $\mathcal{G}_C$ in C and each node $\nu \in \mathcal{G}_C$ there is a graph $\mathcal{G}_G$ in G such that $\mathcal{G}_C\,|_\nu \sqsubseteq \mathcal{G}_G$.*

It follows by the definition that if C is an HPSG corpus and Γ is a corpus grammar of C then Γ accepts all analyses in C.

4. Incremental Specialization using Negative Information

Let us now return to the annotation process. We start with an HPSG grammar which together with the signature determines the annotation scheme. We convert this grammar into a graph representation $\mathcal{GR}_0$. In the project we rely on the existing system (TRALE) for processing of HPSG grammars (TRALE is based on (Götz and Meurers, 1997)). TRALE works with HPSG grammars represented as general descriptions, but the result from the sentence processing is equivalent to a complete feature graph. It is also relatively easy to convert the grammar into a set of feature graphs.

Having $\mathcal{GR}_0$ we can analyze partial analyses of the sentences as it was described in the introduction. The partial analyses are used in order to reduce the number of the possible analyses. Let us suppose that the set of complete feature graphs $\mathcal{GRA}$ is returned by the TRALE system. Then these graphs are processed by the annotator within the CLaRK system and some of the analyses are accepted to be true for the sentence. Thus, they are added to the corpus and the rest of the analyses are rejected. Let $\mathcal{GRN}$ be the set of rejected analyses and $\mathcal{GRC}$ be the set of all analyses in the corpus up to now plus the new accepted ones. Our goal now is to specialize the initial grammar $\mathcal{GR}_0$ into a grammar $\mathcal{GR}_1$ such that it is still a grammar of the corpus $\mathcal{GRC}$ and it does not derive any of the graphs in $\mathcal{GRN}$. Using Prop. 6 we can rely on a very simple test for acceptance or rejection of a complete graph by the grammar: "If for each node in a complete graph there exists a graph in the grammar that subsumes the subgraph started at the same node, then the complete graph is accepted by the grammar." So, in order to reject a graph $\mathcal{G}$ in $\mathcal{GRN}$ it is enough to find a node ν in $\mathcal{G}$ such that for the subgraph $\mathcal{G}\,|_\nu$ there is no graph $\mathcal{G}' \in \mathcal{GR}_1$ such that $\mathcal{G}\,|_\nu \sqsubseteq \mathcal{G}'$. We will use this dependency in the process of guiding the specialization of the initial grammar.

In order to apply this test we have to consider not only the graphs in $\mathcal{GRC}$ and $\mathcal{GRN}$, but also their complete subgraphs. We process further the graphs in $\mathcal{GRN}$ and $\mathcal{GRC}$ in order to determine which information encoded in these graphs is crucial for the rejection of the graphs in $\mathcal{GRN}$. Let $sub(\mathcal{GRN})$ be the set of the complete graphs in $\mathcal{GRN}$ and their complete subgraphs and let $sub(\mathcal{GRC})$ be the set of the complete graphs in $\mathcal{GRC}$ and their complete subgraphs. We divide the set $sub(\mathcal{GRN})$ into two sets: $\mathcal{GRN}^+$ and $\mathcal{GRN}^-$, where $\mathcal{GRN}^+ = sub(\mathcal{GRN}) \cap sub(\mathcal{GRC})$ contains all graphs that are equivalent to some graph as well in $\mathcal{GRP}$[3] and $\mathcal{GRN}^- = sub(\mathcal{GRN}) \setminus sub(\mathcal{GRC})$ contains subgraphs that are presented only in $sub(\mathcal{GRN})$.

Then we choose all graphs $\mathcal{G}$ in $\mathcal{GR}_0$ such that for some $\mathcal{G}' \in \mathcal{GRN}^-$ it holds $\mathcal{G}' \sqsubseteq \mathcal{G}$. Let this set be $\mathcal{GR}_0^-$. This is the set of graphs in the grammar $\mathcal{GR}_0$ which we have to modify in order to achieve our goal.

Then we select from $sub(\mathcal{GRC})$ all graphs such that they are subsumed by some graph from $\mathcal{GR}_0^-$. Let this set be $\mathcal{GRP}$. These are the graphs that might be rejected by the modified grammar. Thus, the algorithm has to disallow such a rejection.

Thus our task is to specialize the graphs in the set $\mathcal{GR}_0^-$ in such a way that the new grammar (after substitution of $\mathcal{GR}_0^-$ with the new set of more specific graph into $\mathcal{GR}_0$) accepts all graphs in $\mathcal{GRP}$ and rejects all graphs in $\mathcal{GRN}$.

The algorithm works by performing the following steps:

[3]This is based on the fact that the accepted analyses can share some subgraphs with the rejected analyses.

1. It calculates the set $\mathcal{GRN}^-$;
2. It selects a subset $\mathcal{GR}_0^-$ of $\mathcal{GR}_0$;
3. It calculates the set $\mathcal{GRP}$;
4. It tries to calculate a new set of graphs $\mathcal{GR}_1^-$ such that each graph $\mathcal{G}$ in the new set $\mathcal{GR}_1^-$ is either member of $\mathcal{GR}_0^-$ or it is subsumed by a graph in $\mathcal{GR}_0^-$. Each new graph in $\mathcal{GR}_1^-$ can not have more nodes than the nodes in the biggest graph in the sets $\mathcal{GRP}$ and $\mathcal{GRN}$. This condition ensures the algorithm termination. If the algorithm succeeds to calculate a new set $\mathcal{GR}_1^-$ then it proceeds with the next step. Otherwise it stops without producing a specialization of the initial grammar.
5. It checks whether each graph in $\mathcal{GRP}$ is subsumed by a graph in $\mathcal{GR}_1^-$. If 'yes' then it prolongs the execution with the next step. Otherwise it returns to step 4 and calculates a new set $\mathcal{GR}_1^-$.
6. It checks whether there is a graph in $\mathcal{GRN}$ such that it is subsumed by a graph in $\mathcal{GR}_1^-$ and all its complete subgraphs in $\mathcal{GRN}^-$ are subsumed by a graph in $\mathcal{GR}_1^-$. If 'yes' then it returns to step 4 and calculates a new set $\mathcal{GR}_1^-$. Otherwise it returns the set $\mathcal{GR}_1^-$ as a specialization of the grammar $\mathcal{GR}_0^-$.

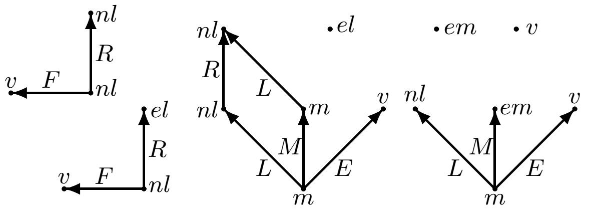
When the algorithm returns a new set of graphs $\mathcal{GR}_1^-$ which is a specialization of the graph set $\mathcal{GR}_0^-$, then we substitute the graph set $\mathcal{GR}_0^-$ with $\mathcal{GR}_1^-$ in the grammar $\mathcal{GR}_0$ and the result is a new, more specific grammar $\mathcal{GR}_1$ such that it accepts all graphs in the corpus $\mathcal{GRC}$ and rejects all graphs in $\mathcal{GRN}$.

In general, of course, there exist more than one specialization. Deciding which one is a good one becomes a problem, which cannot be solved only on the base of the graphs in the two sets $\mathcal{GRP}$ and $\mathcal{GRN}$. In this case two repairing strategies are possible: either additional definition of criteria for choosing the best extension, or the application of some statistical evaluations.

If the algorithm fails to produce a new set of graphs $\mathcal{GR}_1^-$ then there is an inconsistency in the acceptance of the graphs in $\mathcal{GRC}$ and/or in the rejection of the graphs in $\mathcal{GRN}$. This could happen if the annotator marks as wrong an analysis (or a part of it) which was marked as true for some previous analysis.

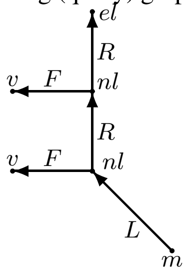
5. Example

In this section we present an example. This example is based on the notion of list and member relation encoded as feature graphs. The lists are encoded by two species: nl for non-empty lists and el for empty lists. Two features are defined for non-empty lists: F for the first element of the list and R for the rest of the list. The elements of a list are of species v. The member relation is also encoded by two species: m for the recursive step of the relation and em for the non-recursive step. For the recursive step of the relation (species m) three features are defined: L pointing to the list, E for the element which is a member of the list and M for the next step in the recursion of the relation. The next set of graphs constitutes an incomplete grammar for member relation on lists. The incompleteness results from the fact that there is no restriction on the feature E.



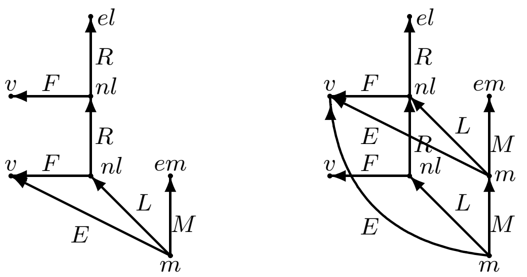
Here the two graphs on the left represent the fact that the rest of a non-empty list could be a non-empty list or an empty list. They also state that each non-empty list has a value. Then there are two graphs for the species m. The first states that the relation member can have a recursive step as a value for the feature M if and only if the list of the second recursive step is the rest of the list of the first recursive step. The second graph just completes the appropriateness for the species m saying that the value of the feature L is also of species non-empty list when the value of the feature M is non-recursive step of the member relation. There are also three graphs with single nodes for the case of empty lists, non-recursive steps of member relations and for the values of the lists. They are presented at the top right part of the picture. Now let us suppose that the annotator would like to enumerate all members of a two-element list by evaluation of the following (query) graph with respect to the above grammar.

Query graph:



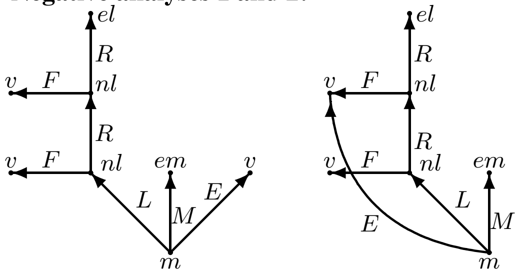
The grammar returns two acceptable analyses. One for the first element of the list and one for the second element of the list.

Positive analyses:

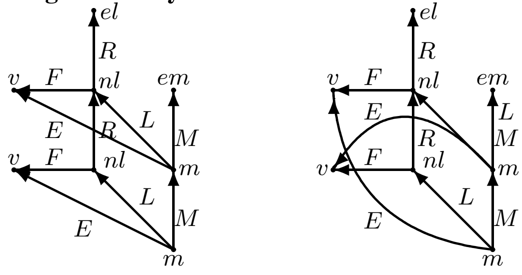


The grammar also accepts 11 wrong analyses in which the E features either point to wrong elements of the list or they are not connected with element of the list at all. Here are the wrong analyses.

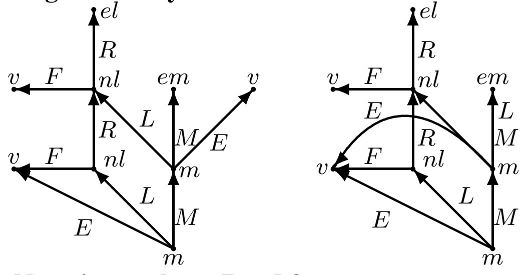
Negative analyses 1 and 2:



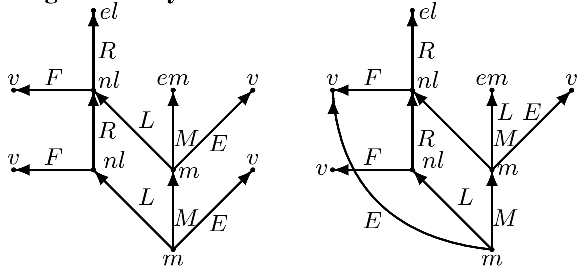
Negative analyses 3 and 4:



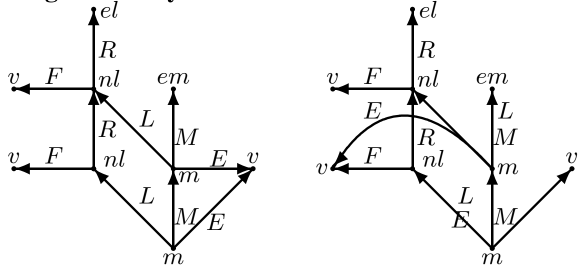
Negative analyses 5 and 6:



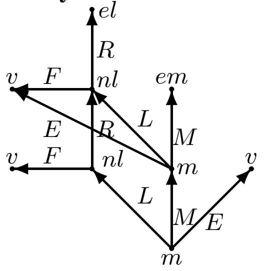
Negative analyses 7 and 8:



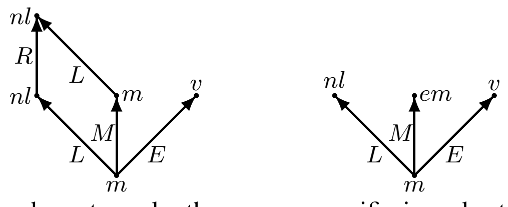
Negative analyses 9 and 10:



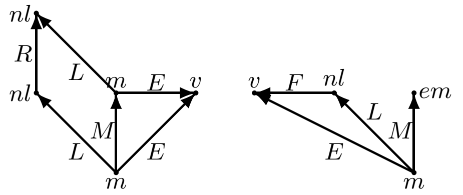
Negative analysis 11:



The next step is to determine the set $\mathcal{GRN}^-$. This set contains 12 complete graphs: all graphs in the set $\mathcal{GRN}$ and one subgraph that is not used in the positive analyses. We will not list these graphs here. The graphs from the grammar that subsumes the graphs in $\mathcal{GRN}^-$ are the two graphs for the member relation. We repeat them here.



Now we have to make them more specific in order to reject the negative examples from $\mathcal{GRN}^-$ but still to accept the two positive examples. The next two graphs are an example of such more specific graphs.



By the first graph the negative examples 3, 4, 5, 7, 8, 10 and 11 are rejected, and by the second graph the negative examples 1, 2, 5, 6, 7, 8, 9, 10 are rejected. Thus both specializations are necessary in order to reject all negative examples. The new grammar still accepts the two positive examples.

6. Conclusion

The presented approach is still very general. It defines a declarative way to improve an annotation HPSG grammar represented as a set of feature graphs. At the moment we have implemented only partially the connection between TRALE system and CLaRK system. Thus, a demonstration of the practical feasibility of the approach remains for future work.

Similar approach can be established on the base of the positive information only (see (Simov, 2001) and (Simov, 2002)), but the use of the negative information can speed up the algorithm. Also, the negative as well as positive information can be used in creation of a performance model for the new grammar along the lines of (Bod, 1998).

7. Acknowledgements

The work reported here is done within the BulTreeBank project. The project is funded by the Volkswagen Stiftung, Federal Republic of Germany under the Programme *"Cooperation with Natural and Engineering Scientists in Central and Eastern Europe"* contract I/76 887.

We would like to thank Petya Osenova for her comments on earlier versions of the paper. All errors remain ours, of course.

8. References

Rens Bod. 1998. *Beyond Grammar: An Experience-Based Theory of Language*. CSLI Publications, CSLI, California, USA.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science 32. Cambridge University Press.

T. Götz and D. Meurers. 1997. *The ConTroll system as large grammar development platform*. In *Proceedings of the ACL/EACL post-conference workshop on Computational Environments for Grammar Development and Linguistic Engineering*. Madrid, Spain.

P.J. King. 1989. *A Logical Formalism for Head-Driven Phrase Structure Grammar*. Doctoral thesis, Manchester University, Manchester, England.

P.J. King. 1999. *Towards Thruth in Head-Driven Phrase Structure Grammar*. In V. Kordoni (Ed.), *Tübingen Studies in HPSG,* Number 132 in Arbeitspapiere des SFB 340, pp 301-352. Germany.

P. King and K. Simov. 1998. The automatic deduction of classificatory systems from linguistic theories. In *Grammars*, volume 1, number 2, pages 103-153. Kluwer Academic Publishers, The Netherlands.

P. King, K. Simov and B. Aldag. 1999. The complexity of modelability in finite and computable signatures of a constraint logic for head-driven phrase structure grammar. In *The Journal of Logic, Language and Information*, volume 8, number 1, pages 83-110. Kluwer Academic Publishers, The Netherlands.

C.J. Pollard and I.A. Sag. 1987. *Information-Based Syntax and Semantics*, vol. 1. CSLI Lecture Notes 13. CSLI, Stanford, California, USA.

C.J. Pollard and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois, USA.

C.J. Pollard. 1999. *Strong Generative Capacity in HPSG.* in Webelhuth, G., Koenig, J.-P., and Kathol, A., editors, *Lexical and Constructional Aspect of Linguistic Explanation,* pp 281-297. CSLI, Stanford, California, USA.

K. Simov. 2001. *Grammar Extraction from an HPSG Corpus.* In: Proc. of the RANLP 2001 Conference, Tzigov chark, Bulgaria, 5–7 Sept., pp. 285–287.

K. Simov, G. Popova, P. Osenova. 2001. *HPSG-based syntactic treebank of Bulgarian (BulTreeBank).* In: *"A Rainbow of Corpora: Corpus Linguistics and the Languages of the World",* edited by Andrew Wilson, Paul Rayson, and Tony McEnery; Lincom-Europa, Munich, pp. 135–142.

K. Simov, Z. Peev, M. Kouylekov, A. Simov, M. Dimitrov, A. Kiryakov. 2001. *CLaRK - an XML-based System for Corpora Development.* In: Proc. of the Corpus Linguistics 2001 Conference, pages: 558-560.

K. Simov. 2002. *Grammar Extraction and Refinement from an HPSG Corpus.* In: Proc. of ESSLLI-2002 Workshop on Machine Learning Approaches in Computational Linguistics, August 5-9.(to appear)

K.Simov, P.Osenova, M.Slavcheva, S.Kolkovska, E.Balabanova, D.Doikoff, K.Ivanova, A.Simov, M.Kouylekov. 2002. *Building a Linguistically Interpreted Corpus of Bulgarian: the BulTreeBank.* In: Proceedings from the LREC conference, Canary Islands, Spain.

A Bootstrapping Approach to Automatic Annotation of Functional Information to Adjectives with an Application to German

Bernd Bohnet, Stefan Klatt and Leo Wanner

Computer Science Department
University of Stuttgart
Breitwiesenstr. 20-22
70565 Stuttgart, Germany
{bohnet|klatt|wanner}@informatik.uni-stuttgart.de

Abstract

We present an approach to automatic classification of adjectives in German with respect to a range functional categories. The approach makes use of the grammatical evidence that (i) the functional category of an adjectival modifier determines its relative ordering in an NP, and (ii) only modifiers that belong to the same category may appear together in a coordination. The coordination context algorithm is discussed in detail. Experiments carried out with this algorithm are described and an evaluation of the experiments is presented.

1. Introduction

Traditionally, corpora are annotated with POS, syntactic structures, and, possibly, also with word senses. However, for certain word categories, further types of information are needed if the annotated corpora are to serve as source, e.g., for the construction of NLP lexica or for various NLP-applications. Among these types of information are the semantic and functional categories of adjectives that occur as premodifiers in nominal phases (NPs) (Raskin and Nirenburg, 1995). In this paper, we focus on the functional categories such as 'deictic', 'numerative', 'epithet', 'classifying', etc. As is well-known from the literature (Halliday, 1994; Engel, 1988), the functional category of an adjectival modifier in an NP predetermines its relative ordering with respect to other modifiers in the NP in question, the possibility of a coordination with other modifiers, and to a certain extent, also the reading in the given communicative context. Consider, e.g. in German,

(1) *Viele junge kommunale Politiker ziehen aufs Land*
'Many young municipal politicians move to the country side'.
but
**Viele kommunale junge Politiker ziehen aufs Land*
'Many municipal young politicians move to the country side'.

(2) *Viele ehemalige Politiker ziehen aufs Land*
'Many previous politicians move to the country side.'
but
**Ehemalige viele Politiker ziehen aufs Land*
'Previous many politicians move to the country side.'

Jung 'young' and *kommunal* 'municipal', *viele* 'many' and *ehemalig* 'previous' belong to different functional categories, which makes them unpermutable in the above NPs and implies a specific relative ordering: category(*jung*) $<$ category(*kommunal*) and category(*viele*) $<$ category(*ehemalig*). In contrast, *jung* 'young' and *dynamisch* 'dynamic' belong to the same category; they can be permuted in an NP without an impact on the grammaticality of the example:

(3) *Viele junge, dynamische Politiker ziehen aufs Land*
'Many young, dynamic politicians move to the country side'.
and
Viele dynamische, junge Politiker ziehen aufs Land
'Many dynamic, young politicians move to the country side'.

They can also appear in a coordination:

(4) *Viele junge und dynamische Politiker ziehen aufs Land*
'Many young and dynamic politicians move to the country side'.
Viele dynamische und junge Politiker ziehen aufs Land
'Many dynamic and young politicians move to the country side'.

while, e.g., *viele* and *kommunal* cannot:

(5) **Viele junge und kommunale Politiker ziehen aufs Land*
'Many young and municipal Stuttgart politicians move to the country side'.

In such applications as natural language generation and machine translation, it is important to have the function of the adjectives specified in the lexicon. However, as yet, no large lexica are available that would contain this information. Therefore, an automatic corpus-based annotation of functional information seems the most suitable option.

In what follows, we present a bootstrapping approach to the functional annotation of German adjectives in corpora. The next section presents a short outline of the theoretical assumptions we make with respect to the function of adjectival modifiers and their occurrence in NPs and coordination contexts, before in Section 3. the preparatory stage and the annotation algorithms are specified. Section 4. contains then the description of the experiments we carried out in order to evaluate our approach, and Section 5. contains the discussion of these experiments. In Section 6., we give some references to work that is related to ours. In Section 7., finally, we draw some conclusions and outline the directions we intend to take in this area in the future.

2. The Grammatical Prerequisits

Grammarians often relate the default ordering of adjectival modifiers to their semantic or functional categories; see, among others, (Dixon, 1982; Engel, 1988; Dixon, 1991; Frawley, 1992; Halliday, 1994). (Vendler, 1968) motivates it by the order of the transformations for the derivation of the NP in question. (Quirk et al., 1985) state that the position of an adjective in an NP depends on how inherent this adjective's meaning is: adjectives with a more inherent meaning are placed closer to the noun than those with a less inherent meaning. (Seiler, 1978) and (Helbig and Buscha, 1999) argue that the order is determined by the scope of the individual adjectival modifiers in an NP. For an overview of the literature on the topic, see, e.g., (Raskin and Nirenburg, 1995).

As mentioned above, we follow the argumentation that the order of adjectives in an NP is determined by their functional categories. In this section, we first outline the range of functions of adjectival modifiers known from the literature especially for German, present then the function-dependent default ordering, and discuss, finally, the results of an empirical study carried out to verify the theoretical postulates and thus to prepare the grounds for the automatic functional category annotation procedure.

2.1. Ranges of Functions of Adjectival Modifiers

In the literature, different ranges of functional categories of adjectival premodifiers have been discussed. For instance, (Halliday, 1994), proposes for English the following categories of the elements in an NP that precede the noun:

(i) deictic: *this*, *those*, *my*, *whose*,...;

(ii) numerative: *many*, *second*, *preceding*, ...;

(iii) epithet: *old*, *blue*, *pretty*, ...;

(iv) classifier: *electric*, *catholic*, *vegetarian*, *Spanish*, ...

In (Engel, 1988), a slightly different range of categories is given for German adjectival premodifiers:

(i) quantitative: *viele* 'many', *einige* 'some', *wenige* 'few', ...

(ii) referential: *erst* 'first', *heutige* 'today's', *diesseitige* 'from-this-side', ...

(iii) qualificative: *schön* 'beautiful', *alt* 'old', *gehoben* 'upper', ...

(iv) classifying: *regional* 'regional', *staatlich* 'state', *katholisch* 'catholic', ...

(v) origin: *Stuttgarter* 'from-Stuttgart', *spanisch* 'Spanish', *marsianisch* 'from-Mars', ...

The function of a modifier may vary with the context of the NP in question or even be ambiguous (Halliday, 1994; Tucker, 1995). Thus, Ger. *zweit* 'second' belong to the referential category in the NP *zweiter Versuch* 'second attempt'; in *zweiter Preis* 'second price', it belongs to the classifying category. *Fast* in *fast train* can be considered as qualificative or as classifying (if *fast train* means 'train classified as express').

Two modifiers are considered to belong to the same category if they can appear together in a coordination or can be permutated in an NP:

(6) a. Ger. *eine rote oder weiße Rose*
'a red or a white rose'

b. *dritter oder vierter Versuch*
'third or fourth attempt'

c. *elektrische oder mechanische Schreibmaschine*
'an electric or mechanic typewriter'

but not

(7) a. $^{??}$*eine rote und langstielige Rose*
'a red and long-stemmed rose'

b. **rote und holländische Rosen*
'red and Dutch roses'

c. **eine schöne oder elektrische Schreibmaschine*
'a beautiful or electric typewriter'

The credibility of the coordination test is limited, however. Consider

(8) $^{??}$ *Eine schöne und rote Rose*
'a beautiful and red rose'

where *schön* 'beautiful' and *rot* 'red' both belong to the qualitative category, but still do not permit a coordination easily.

Adjectival modifier function taxonomies are certainly language-specific (Frawley, 1992). Nonetheless, as the taxonomies suggested by Halliday and Engel show, they may overlap to a major extent. Often, the difference is more of a terminological than of a semantic nature. In our work, we adopt Engel's taxonomy.

2.2. The Default Ordering of Adjectival Modifiers

Engel (Engel, 1988) suggests the following default ordering of modifier functions:
quantitative < referential < qualificative < classifying < origin

Cf, e.g.:

quant.	referent.	qual.	class.	origin
viele 'many'	ehemalige 'previous'	junge 'young'	kommunale 'municipal'	Stuttgarter 'Stuttgart'

as in

(9) *Viele ehemalige junge kommunale Stuttgarter Politiker ziehen aufs Land*
'Many previous young municipal Stuttgart politicians move to the country side'.

According to Engel, a violation of this default ordering leads to ungrammatical NPs. (1–3) in the Introduction illustrate this violation.

2.3. Empirical Evidence for the Theoretical Claims

In the first stage of our work, we sought empirical evidence for the theoretical claims with respect to the functional category motivated ordering and the functional category motivated coordination restrictions. Although, in general, these claims have been buttressed by our study, counterexamples were found in the corpus with respect to both of them.

2.3.1. Default Ordering: Counterexamples

Especially adjectives of the category 'origin' tended to occur before classifying or qualificative modifiers instead of being placed immediately left to the noun–as would be required by the default ordering. For instance, *spanisch* 'Spanish' occured in 3.5% of its occurrences in the corpus in other positions; cf., for illustration:

(10) a. (*das*) *spanische höfische Bild*
'(the) Spanish courtly picture'

b. (*der*) *spanische schwarze Humor*
'(the) Spanish black humour'

c. (*der*) *spanischen sozialistischen Partei*
'(the) Spanish socialist party$_{\mathrm{dat}}$'

To be noted is that in such NPs as (*der*) *spanische schwarze Humor* and *deutsche katholische Kirche* 'German catholic church' the noun and the first modifier form a multiword lexeme rather than a freely composed NP (i.e. *schwarzer Humor* 'black humour' and *katholische Kirche* 'catholic church'). That is, the preceding modifiers (*spanisch* 'Spanish'/*deutsch* 'German') function as modifiers of the respective multiword lexeme, not of the noun only. This is also in accordance with (Helbig and Buscha, 1999)'s scope proposal.

2.3.2. Coordination Restrictions: Counterexamples

It is mainly ordinals that occur, contrary to the theoretical claim, in coordinations with modifiers that belong to a different category. For instance, *erst* 'first' appears in the corpus in 9.74% cases of its occurrence in such "heterogeneous" coordinations. Cf., for illustration:

(11) a. (*die*) *erste und wichtigste Aufgabe*
'(the) first and the most important task'

b. (*eines der*) *ersten und augenfälligsten Projekte*
'one of the first and conspicuous projects'

c. (*die*) *oberste und erste Pflicht*
'(the) supreme and first duty'

As a rule, in such cases the ordinals have a classifying function, which is hard to capture, however.

2.3.3. Grammaticality of the Counterexamples

An evaluation of the counterexamples found in the corpus revealed that not all of these examples can, in fact, be considered as providing counter evidence for the theoretical claims. The grammaticality of a considerable number of these examples has been questioned by several speakers of German; cf., for instance:

(12) a. *(*die*) *ersten und fehlerhaften Informationen*
'(the) first and erroneous informations'

b. $^{??}$*jüngster und erster Präsident*
'youngest and first president'

c. $^{??}$(*die*) *oberste und erste Pflicht*
'(the) supreme and first duty'

3. The Approach

The empirical study of the relative ordering of adjectival modifiers in NPs and of adjectival modifier coordinations in the corpus showed that the theoretical claims made with respect to the interdependency between functional categories and ordering respectively coordination context restrictions are not always proved right. However, deviances from these claims encountered are not numerous enough to question these claims. Therefore, in our approach to the automatic annotation of adjectival modifiers in NPs with functional information outlined below, we make use of them.

The basic idea underlying the approach can be summarized as follows:

1. take a small set of samples for each functional category as point of departure;
2. look in the corpus for coordinations in which one of the elements is in the starting set (and whose functional category is thus known) and the other element is not yet annotated and annotate it with the category of the first element;

   alternatively:

   look in the corpus for all NP-contexts in which one of the elements is in the starting set, assign to its left and right neighbors all categories that these can may have according to the default ordering;
3. attempt to further constrict the range of categories of all modifiers that are still assigned more than one category;
4. add the unambiguously annotated modifers to the set of samples and repeat the annotation procedure;
5. terminate if all adjectival modifiers have been annotated a unique functional category or no further constrictions are possible.

Note that we do not take the punctuation rule into account, which states that adjectival modifiers of the same category are separated by a comma, while modifiers of different categories are not separated. This is because this rule is considered to be unreliable in practice. Furthermore, we do not use such hints as that classifying modifiers do not appear in comparative and superlative forms. See, however, Section 7.

3.1. The Preparatory Stage

The preparatory stage consists of three phases: (i) preprocessing the corpus, (ii) pre-annotation of modifiers whose category is *a priori* known, and (iii) compilation of the sets of modifiers from which the annotation algorithms start.

3.1.1. Preprocessing the Corpus

To have the largest possible corpus at the lowest possible cost, we start with a corpus that is not annotated with POS. When preprocessing the corpus, first token sequences are identified in which one or several tokens with an attributive adjectival suffix (*-e*, *-es*, *-en*, *-er*, or *-em*) are written in small letters and are followed by a capitalized token assumed to be a noun.[1] The tokens with an attributive suffix may be separated by a blank, a comma or have the conjunction *und* 'and' or the disjunction *oder* 'or' in between: cf.:

(13) a. (*das*) *erste richtige Beispiel*
'(the) first correct example'

b. *rote, blaue und grüne oder schwarze Hosen*
'red, blue and green or black pants'

Note that this strategy does not capture certain marginal NP-types; e.g.:

(a) NPs with an irregular adjectival suffix; e.g., *-a*: (*eine*) *lila Tasche* '(a) purple bag', *rosa Haare* 'pink hair', etc.;

(b) NPs with adjectival modifiers that start with a capital.

However, NPs of type (a) are very rare and can more reliably be annotated manually. NPs of type (b) are, first of all, modifiers at the beginning of sentences and attributive uses of proper nouns; cf. *Sorgenloses* 'free of care' in *Sorgenloses Leben – das ist das, was ich will!* lit. 'Free-of-care life—this is what I want' and *Franfurter* 'Frankfurt' in *Frankfurter Würstchen* 'Frankfurt sausages'. The first type appears very seldom in the corpus and can thus be neglected; for the second type, other annotation strategies proved to be more appropriate (Klatt, forthcoming).

After the token sequence identification, wrongly selected sequences are searched for (cf., e.g., *eine schöne Bescherung* 'a nice mess', where *eine* 'a' is despite its suffix obviously not an adjective but an article). This is done by using a morphological analysis program.

3.1.2. Pre-Annotation

In the pre-annotation phase, the following tasks are carried out:

- Adjectival modifiers of the category 'quantitative' are manually searched for and annotated. This is because the set of these modifiers is very small (*einige* 'some', *wenige* 'few', *viele* 'many', *mehrere* 'several') and would not justify the attempt of an automatic annotation.

- In (Engel, 1988), ordinals are by default considered to be referential. Therefore, we use a morphological analysis program to identify ordinals in order to annotate them accordingly in a separate procedure.

- Engel considers attributive readings of verb participles to be qualitative. This enables us to annotate participles with the qualitative function tag before the actual annotation algoritm is run.

3.1.3. Compiling the Starting Sets

Once the corpus is preprocessed and the pre-annotation is done, the starting sample sets for the annotation algorithms are compiled: for each category, a starting set of samples is manually chosen. The number of samples in each set is not fixed. In the experiments we carried out to evaluate our approach the size of sets varied from one to four (cf. Tables 3 and 5 below).

3.2. The Annotation Algorithms

The annotation program consists of two algorithms that can be executed in sequence or independently of each other. The first algorithm processes coordination contexts only. The second algorithm processes NP-contexts in general.

3.2.1. The Coordination Context Algorithm

The coordination context algorithm makes use of the knowledge that two adjectival modifiers that appear together in a conjunction or disjunction belong to the same functional category. As mentioned above, it loops over the set of modifiers whose category is already known (at the beginning, this is the starting set) looking for coordinations in which one of the elements is member of this set and the other element is not yet annotated. The element not yet annotated is assigned the same category as carried by the element already annotated.

The algorithm can be summarized as follows:

1. For each starting set in the starting set configuration do:
   (a) Mark each element in the set as `starting element` and as `processed`.
   (b) Retrieve all coordinations in which one of the `starting elements` occurs;
   for the not yet annotated elements in the coordinations do
      – mark each of them as `preprocessed`;
      – annotate each of them with the same category as assigned to its already annotated respective neighbor;
      – make a frequency distribution of them.
   (c) determine the element in the above frequency distribution with the highest frequency that is not marked as `processed` and mark this element as the next iteration `candidate` of the functional category in question.
2. Take the next iteration `candidate` with the highest frequency of the sets of all categories and mark it as `processed`. Stop, if no next iteration `candidate`

[1]Recall that in German nouns are capitalized.

can be found in any of the newly annotated elements of one of the categories.

3. Find all new corresponding coordination neighbors, add these elements to the set of `preprocessed` elements for the given category and make a new frequency distribution.

4. Determine the next iteration `candidate` for the given category as done in step 1c.

5. Continue with step 2.

Note that the coordination context algorithm does not loop over one of the categories a predetermined number of times and passes on then to the next category in order to repeat the same procedure. Rather, the switch from category to category is determined solely on the basis of the frequency distribution: the most frequent modifier not yet annotated is automaticaly chosen for annotation—independently of the category that has been assigned before. This strategy has two major advantages:

- it takes into account that the distribution of the modifiers in the corpus over the functional categories is extremely unbalanced: the set of 'quantitatives' counts only a few members while the set of 'qualitatives' is very large.

- it helps avoid an effect of "over-annotation" in the course of which the choice of an element that has already been selected as next iteration `candidate` for a specific category as next iteration `candidate` for a different category would lead to a revision of the annotation of all other already annotated elements involved in coordinations with this element.

Especially the second advantage contributes to the quality of our annotation approach. However, obviously enough, this algorithm assigns only one functional category to each adjective. That is, a multiple category assignment that is desirable in certain contexts must be pursued by another algorithm. This is done by the NP-context algorithm discussed in the next subsection.

Table 1 shows a few iterations of the coordination context algorithm with the starting sets of Experiment 1 in Section 4.. Here and henceforth the functional categories are numbered as follows:

1	2	3	4	5
↓	↓	↓	↓	↓
quant.	referent.	qualificat.	class.	origin

In the first iteration, the most frequent "next iteration candidate" of category 1 is *solch* 'such'with a frequency of 10, the most frequent of category 2 is *letzt* 'last' with a frequency of 71, and so on. The candidate of category 4 *wirtschaftlich* 'economic' possesses the highest frequency; therefore it is chosen for annotation and taken as "next iteration starting element" (see Step 2 in the algorithm outline). After adding all elements that occur in a coordination with *wirtschaftlich* to the candidate list, in iteration 2 the next element for annotation (and thus also the starting element) is chosen. This is done as described above for Iteration 1.

It.	cat.1	cat.2	cat.3	cat.4	cat.5
1	solch (10)	letzt (71)	klein (195)	wirtschaftlich (350)	französisch (93)
2	solch (10)	letzt (71)	klein (195)	sozial (295)	französisch (93)
3	solch (10)	letzt (71)	klein (195)	kulturell (208)	französisch (93)
4	solch (10)	letzt (71)	klein (195)	gesellschaftlich (119)	französisch (93)
5	solch (10)	letzt (71)	mittler (370)	gesellschaftlich (119)	französisch (93)
6	solch (10)	letzt (71)	alt (84)	gesellschaftlich (119)	französisch (93)
7	solch (10)	letzt (71)	alt (84)	ökonomisch (105)	französisch (93)
8	solch (10)	letzt (71)	alt (84)	ökologisch (118)	französisch (93)
9	solch (10)	letzt (71)	alt (84)	militärisch (74)	französisch (93)
10	solch (10)	letzt (71)	alt (84)	militärisch (74)	amerikanisch (95)

Table 1: An excerpt of the first iterations of the coordination context algorithm

3.3. The NP-Context Algorithm

The NP-context algorithm is based on the functional category motivated relative ordering of adjectival modifiers in an NP as proposed by Engel (see Section 2.).

In contrast to the coordination-context algorithm, which always ensures a non-ambiguous determination of the category of an adjective, the NP-context algorithm is more of an auxiliary nature. It helps to (i) identify cases where an adjective can be assigned multiple categories, (ii) make hypotheses with respect to categories of adjectival modifiers that do not appear in coordinations, (iii) verify the category assignment of the coordination-context algorithm.

The NP-context algorithm allows for a non-ambiguous determination of the category only in the case of a "comprehensive" NP, i.e., when all positions of an NP (from 'quantitative' to 'origin' are instantiated. Otherwise, relative statements of the kind as in the following case are possible:

> Given the NP (*der*) *schöne, junge, grüne Baum* '(the) beautiful, young, green tree', from which we know that *jung* 'young' is qualitative, we can conclude that *schön* may belong to one of the following three categories: quantitative, referential, or also qualitative, and that *grün* is either qualitative or classifying.

In other words, the following rules underlie the NP-context algorithm:
Given an adjective in an NP whose category X is known:

- assign to all left neighbors of this adjective the categories Y with $Y = 1, 2, \ldots, X$ (i.e., all categories with the number $\leq X$)

- assign to all right neighbors of this adjective the categories Z with $Z = X, X+1, \ldots, 5$ (i.e., all categories with the number $\geq X$

The NP-context algorithm varies slightly depending on the task it is used for—the verification of the categories assigned by the coordination-context algorithm or putting forward hypotheses with respect to the category of adjectives. When being used for the first task, it looks as follows:

1. for all adjectives that received a category tag during the coordination-context algorithm do
   - overtake this tag for all instances of these adjectives in the NP-contexts
2. do for each `candidate` that has been annotated a category
   - for each of the five categories C do
     - assign tentatively C to `candidate`
     - evaluate the NP-context of `candidate` as follows:

       (a) if the other modifiers in the context do not possess category tags, mark the context as unsuitable for the verification procedure

       (b) else, if with respect to the numerical category labels (see above) there is a decreasing pair of adjacent labels (i.e. of neighbor adjectives), mark this NP-context as rejecting C as category of `candidate`, otherwise mark the NP-context as accepting C as category of `candidate`
3. Choose the category whose choice received the highest number of confirmative coordination contexts

Table 2 shows the result of the verification of the category of a few adjectives. The first column contains the adjective whose category is verified. The second column contains the numerical category labels; with a '+' the category prognosticated by the coordination-context algorithm is marked.[2] In the third column, the number of confirmations of the corresponding category by NP-contexts is indicated (i.e. in the case of *neu* 'new', 6083 NP-contexts confirm category 3 ('qualificative') of *neu*, 5048 confirm category 4 ('classifying'), etc.). In the fourth column, the number of NP-contexts is specified that do not provide any evidence for the corresponding category. And in the fifth column the number of NP-contexts is indicated that negate the corresponding function. For four adjectives in Table 2 (*neu* 'new', *groß* 'big', *finanziell* 'financial', and *bosnisch* 'Bosnian') the NP-context algorithm confirmed the category suggested by the coordination-context algorithm; for two adjectives different categories were suggested (for *deutsch* 'German' 4 (classifying) instead of 5 (origin) and for *politisch* 'political' 5 instead of 4).

In the current version of the NP-context algorithm, for adjectival modifiers of category 4 or 5, the correct category

[2]In all six cases, the coordination-context algorithm assignment was correct.

neu	+3	6083	697	112
	4	5048	697	1147
	2	4289	697	1906
	1	4195	697	2000
	5	3360	697	2835
groß	+3	6015	353	74
	2	5314	353	775
	4	5070	353	1019
	1	4391	353	1698
	5	3634	353	2455
deutsch	4	4992	498	109
	+5	4933	498	168
	3	4911	498	190
	2	1111	498	3990
	1	397	498	4704
politisch	5	3615	253	11
	+4	3519	253	107
	3	3353	253	273
	2	267	253	3359
	1	160	253	3466
finanziell	+4	1322	130	1
	5	1321	130	2
	3	1310	130	13
	2	46	130	1277
	1	25	130	1298
bosnisch	+5	223	24	2
	4	217	24	8
	3	214	24	11
	2	17	24	208
	1	11	24	214

Table 2: Examples of categorial classification by the NP-context algorithm

is quite often listed as the second best choice. To avoid an incorrect annotation, further measures need to be taken (see also Section 7.).

4. Experiments with the Coordination Algorithm

To evaluate the performance of the algorithms suggested in the previous section, we carried out experiments in two phases, three experiments each phase. The phases varied with respect to the size of the corpora used; the experiments in each phase varied with respect to the size of the starter sets.

In what follows, the experiments with the coordination algorithm only are discussed.

4.1. The Data

The experiments of the first phase were run on the *Stuttgarter-Zeitung* (STZ) corpus, which contains 36 Mio tokens; the experiments of the second phase were run on the corpus that consisted of the STZ-corpus and the *Frankfurter Rundschau* (FR) corpus with 40 Mio tokens; cf. Table 3. The first row in Table 3 shows the number of adjectival modifier coordinations and the number of premodifier NPs without coordinations in the STZ-corpus and in the STZ+FR-corpus; the second row shows the number of different adjectives that occur in all of these constructions in the respective corpus.

	STZ		STZ+FR	
	coord	NP	coord	NP
# contexts	18648	67757	36985	120673
# diff. adjectives	5894	10035	8003	12993

Table 3: Composition of the adjectival premodifier contexts in our corpora

		number of adjectival mods.						
exp	type	2	3	4	5	6	7	$\sum$
1-3	coord	17228	1238	149	31	2		18648
1-3	NP	66692	1059	6				67757
4-6	coord	34035	2598	298	47	6	1	36985
4-6	NP	118886	1772	15				120673

Table 4: Statistics on the size of the adjectival groups in STZ and STZ+FR

This gives us a ratio of 6.7 between the number of NPs and the number of different adjectives (i.e., the average number of NPs in which a specific adjective occurs) for the STZ-corpus and a ratio of 10.0 for the STZ+FR-corpus. Not surprisingly, larger corpora show a higher adjective repetition rate than small corpora do.

Table 4 contains the statistics on the size of modifier coordinations and the number of adjectival modifiers in NPs in general across both of our corpora. Adjectival modifier groups of size 3 or greater were thus very seldom.

Table 5 contains the data on the composition of both corpora with respect to ordinals and participles of which we assume to know *a priori* to which category they belong: ordinals to the category 2 (‘referential’) and participles to the category 3 (‘qualitative’); see Section 2.

The starter sets consisted for the experiments 1 and 4 of one sample per category: an adjectival modifier of the corresponding category with a high frequency in the STZ-corpus. For the experiments 2 and 5, two, respectively three, high frequency samples for each category were added to starter sets. For the experiments 3 and 6, the starter sets were further extended by an additional modifier which has been assigned a wrong category in the experiments before. Table 6 shows the composition of the starter sets used for the experiments.

Apart from these “regular” members of the starter sets, to the starter sets of category 2 all ordinals and to the starter sets of category 3 all participles available in the respective corpus were added.

To have reliable data for the evaluation of the performance of the annotation program, we let an independent expert annotate 1000 adjectives with functional category

	STZ		STZ+FR	
	ordin.	part.	ordin.	part.
# diff. modifs	24	2023	25	2851
# total occur.	914	5135	2291	10045

Table 5: The distribution of ordinals and participles in STZ and STZ+FR

exp.	cat.1	cat.2	cat.3	cat.4	cat.5
1/4	ander	heutig	groß	politisch	deutsch
2/5	ander solch	heutig letzt einzig	groß alt rot	politisch demokratisch kommunal	deutsch amerikanisch französisch
3/6	ander solch	heutig letzt einzig mittler	groß alt rot schön	politisch demokratisch kommunal katholisch	deutsch amerikanisch französisch russisch

Table 6: The composition of the starter sets

exp.	in total	assigned	¬assigned	p (%)
1	5894	5515	379	82.90%
2	5894	5515	379	84.30%
3	5894	5515	379	84.44%

Table 7: Results of the experiments 1 to 3

information. The manually annotated data were then compared with the output of our program to estimate the precision figures (see below).

4.2. Phase 1 Experiments

In the experiments 1 to 3, we were able to assign a functional category to 93,6% of the adjectival modifiers with all three starter sets. In 379 cases, the program could not assign a category; we discuss these cases in Section 5.. Table 7 summarizes the results of the experiments 1 to 3 (‘p’ stands for “precision”).

Many of the 1000 manually annotated tokens occur only a few times in the corpus (and appear thus in a few coordinations). Low frequency tokens negatively influence the precision rate of the algorithm. The diagrams in Figures 1 to 3 illustrate the number of erroneous annotations in the experiments 1 to 3 in relation to the number of coordinations in which a token chosen as next for annotation appears as element at the moment when *n* tokens from the manually annotated token set have already been annotated. For instance, in Experiment 1, the first time when less than or 100 coordinations are considered to determine the category of a token, 9 of the 1000 members of the test set were annotated correctly, the first time when less than or 75 coordinations are considered, 17 of 1000 received the correct category, the first time when less than or 50 coordinations are considered, 31 tokens received the correct category and one a wrong one. And so on. Note, when less than or 5 coordinations were considered for the first time, only 41 annotations (out of 565) were wrong. This gives us a precision rate of $((565 - 41)/565) \times 100 = 92.74\%$.

Figures 2 and 3 show the annotation statistics for Experiments 2 and 3. Note that in Experiment 2 the precision rate for high frequency adjectives is considerably better than in Experiment 1: when 5 coordination contexts are available for the annotation decision, only 26 mistakes were made (instead of 41 in Experiment 1). Figure 3 shows that by a further extension of the starter set, no reasonable improvement of the results is achieved.

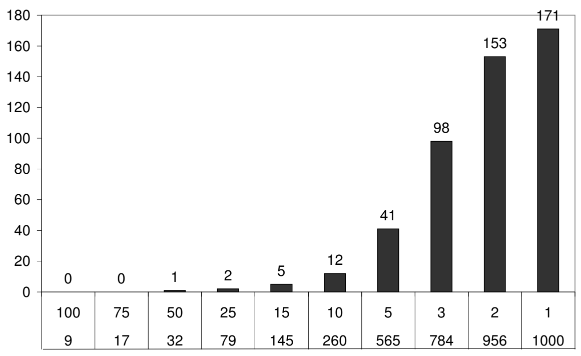


Figure 1: The annotation statistics in Experiment 1

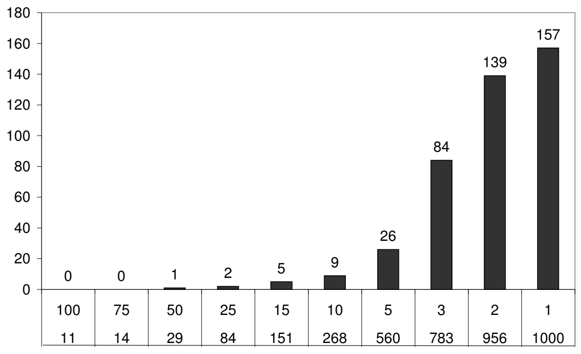


Figure 2: The annotation statistics in Experiment 2

4.3. Phase 2 Experiments

In experiments 4 to 6 we were able to assign with all three starter sets a functional category to 94,1% of the adjectival modifiers, i.e/, to 0.5% more than in the experiments of Phase 1. However, as Table 8 shows, the precision rate decreased slightly. Figures 4 to 6 show the annotation statistics for the Phase 2 experiments.

5. Discussion

In what follows, we first discuss the first 20 iterations of the coordination algorithm in Experiment 1 and Experiment 2, respectively, and present then the overall results of the experiments.

exp.	in total	assigned	¬assigned	p (%)
4	8003	7558	445	84.08 %
5	8003	7558	445	84.08%
6	8003	7558	445	84.92%

Table 8: Results of the experiments 4 to 6

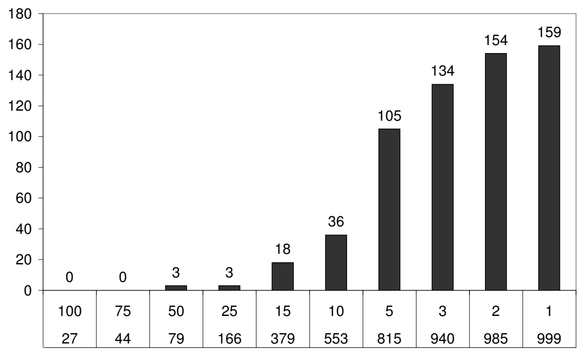


Figure 4: The annotation statistics in Experiment 4

5.1. A Snapshot of the Iterations in Experiments 1 and 2

Table 9 shows the first twenty iterations in Experiment 1, and Table 10 the first twenty iterations in Experiment 2. They look very similar despite the different starting sets in both experiments. Thus, in both nearly the same modifiers are annotated in nearly the same order—except *neu*, which is in Experiment 1 annotated in iteration 14, while in Experiment 2 in iteration 3. At the first glance, one might think that both experiments show the same results. However, as already pointed out above, the bigger starter set in Experiment 2 results in a considerably better precision rates with high and middle frequency adjectives.

5.2. Evaluation of the Experiments

Table 11 shows the distribution of the adjectival modifiers in the six experiments among the five functional categories.

Let us now consider some wrong annotations and some cases where the program was not able to assign a category.

In Table 12, some wrong annotations of category '3' (qualitative) in Experiment 1 are listed. The first column of the table specifies in which iteration of the algorithm the

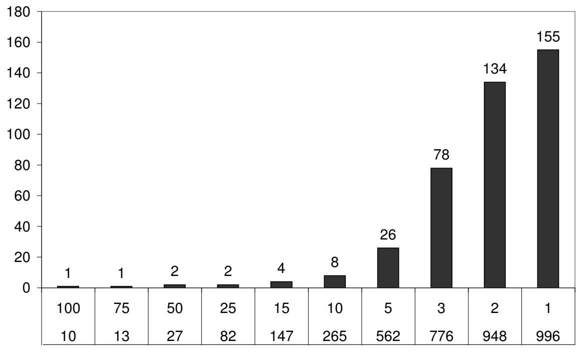


Figure 3: The annotation statistics in Experiment 3

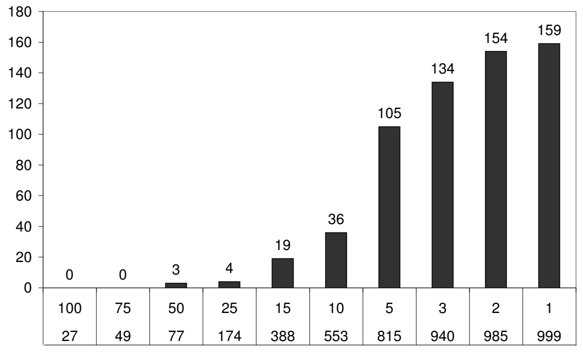


Figure 5: The annotation statistics in Experiment 5

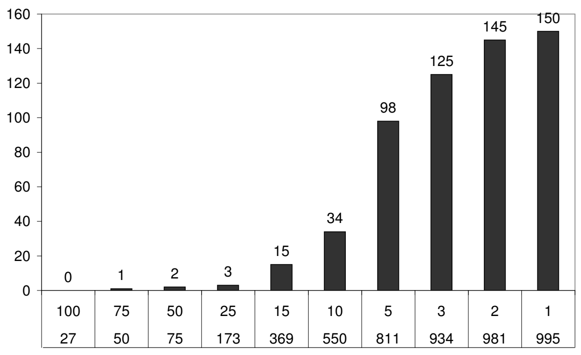


Figure 6: The annotation statistics in Experiment 6

Nr.	adjective	cat.	it. freq	freq
1	wirtschaftlich	4	350	851
2	sozial	4	295	707
3	kulturell	4	208	382
4	klein	3	195	688
5	mittler	3	370	482
6	gesellschaftlich	4	119	178
7	ökonomisch	4	105	167
8	ökologisch	4	118	164
9	französisch	5	93	251
10	amerikanisch	5	95	286
11	europäisch	5	102	179
12	ausländisch	5	88	128
13	alt	3	84	473
14	neu	3	307	417
15	britisch	5	81	100
16	italienisch	5	78	118
17	militärisch	4	74	127
18	letzt	2	71	99
19	finanziell	4	68	264
20	technisch	4	78	258

Table 9: The first 20 iterations in Experiment 1

Nr.	adjective	cat.	it. freq	freq
1	wirtschaftlich	4	356	851
2	sozial	4	307	707
3	neu	3	304	417
4	kulturell	4	210	382
5	klein	3	199	688
6	mittler	3	373	482
7	gesellschaftlich	4	119	178
8	ökonomisch	4	105	167
9	ökologisch	4	119	164
10	europäisch	5	102	179
11	ausländisch	5	88	128
12	britisch	5	81	100
13	italienisch	5	78	118
14	militärisch	4	74	127
15	finanziell	4	68	264
16	technisch	4	78	258
17	religiös	4	67	132
18	englisch	5	67	76
19	jung	3	63	112
20	personell	4	60	141

Table 10: The first 20 iterations in Experiment 2

exp.	cat.1	cat.2	cat.3	cat.4	cat.5	$\sum$
1	8	39	4506	711	251	5515
2	8	39	4434	785	249	5515
3	7	76	4377	791	264	5515
4	13	55	5938	1186	366	7558
5	13	55	5938	1186	366	7588
6	13	63	5926	1200	356	7558

Table 11: Distribution of the adjectival modifiers

Nr.	adjective	cat.	it. freq	freq
64	unter	3	33	43
151	marktwirtschaftlich	3	16	26
780	sozialdemokratisch	3	4	9
782	kommunistisch	3	4	17
807	katholisch	3	5	77
808	evangelisch	3	57	57
809	protestantisch	3	13	17
810	anglikanisch	3	5	5
811	reformerisch	3	4	4

Table 12: Some errors in Experiment 1

respective adjective has been assigned a category. 'it freq' (*iteration frequency*) specifies the number of the coordinations with this adjective as element that were available in the corresponding iteration; 'total freq' specifies how many times the adjective occured in coordinations in the corpus in total.

The correct category of *unter* 'under' would have been 2 ('referential'); that of *marktwirtschaftlich* 'free-enterprise' 4 ('classifying'), that of *kommunistisch* 'communist' 4, etc. Note the case of *katholisch* 'catholic'. Its total frequency of 77 is much higher as that of the adjectives processed before. However, it was chosen with an iteration frequency of only 5, i.e., only 5 coordinations have been considered to determine its category. The consequence is that the following adjectives (cf. iterations 808-811) also received a wrong annotation.

Table 13 shows the first 10 of the 445 adjectives that have not been assigned a category in Experiment 6.

Consider, e.g., the coordination constructions in which, e.g., *neunziger* 'ninety/nineties' occurs: *achtziger* 'eighty/eighties' COORD *neunziger* (11 times) and *siebziger* 'seventy/seventies' COORD *achtziger* COORD *neunziger* (1 time). That is, we run into a deadlock here:

	adjective	freq
1.	sechziger	248
2.	siebziger	195
3.	fünfziger	147
4.	dreißiger	102
5.	achtziger	93
6.	zwanziger	81
7.	vierziger	61
8.	zehner	21
9.	neunziger	12
10.	deutsch-polnisch	6

Table 13: Unprocessed adjectives in Experiment 6

- gradable adj.
  - scalar gradables
    - attitude-based
    - numerical scale
    - literal scale
    - member
  - non-scalar gradables
- non-scalar adj.
  - proper non-scalars
  - event-related non-scalars
  - true relative non-scalars

Figure 7: The taxonomy that underlies the adjective classification by Raskin and Nirenburg

neunziger cannot be assigned a category because all its coordination neighbors did not receive a category either.

6. Related Work

To our knowledge, ours is the first approach to the automatic classification of adjectives with respect to a range of functional categories. In the past, approaches to the classification of adjectives focused on the classification with respect to semantic taxonomies. For instance, (Raskin and Nirenburg, 1995) discuss a manual classification procedure in the framework of the *MikroKosmos*. The taxonomy they refer to is is shown in Figure 7.

Obviously, an automatization of the classification with respect to this taxonomy is still beyond the state of the art in the field. On the other side, (Engel, 1988)'s functional categories seem to suffice to solve, e.g., the problem of word ordering in text generation.

(Hatzivassiloglou and McKeown, 1993) suggest an algorithm for clustering adjectives according to meaning. However, they do not refer to a predetermined (semantic) typology or set of functional categories.

(Hatzivassiloglou and McKeown, 1997) determine the orientation of the adjectives (negative vs. positive). The orientation is a useful lexical information since it has an impact on the use of adjectives in coordinations: only adjectives with the same orientation appear easily in conjunctions; cf. $^{??}$*stupid and pretty* but *stupid but pretty*. So far, we do not annotate orientation information.

(Shaw and Hatzivassiloglou, 1999)'s work explicitly addresses the problem of the relative ordering of adjectives. In contrast to ours, their approach suggests a pairwise relative ordering of concrete adjectives, not of functional or semantic categories.

7. Conclusions and Future Work

We presented two simple algorithms for the classification of adjectives with respect to a range of functional categories. One of these algorithms, the *coordination context* algorithm, has been discussed in detail. The precision rate achieved by this algorithm is encouraging. It is better for high frequency adjectives than for low frequency adjectives.

Our approach can be considered as a first step into the right direction. In order to achieve better results, we intend to extend our approach along two lines:

- incorporation of additional linguistic clues (e.g., that classifier modifiers do not appear in comparative and superlative forms, that modifiers of the same category can be separated by a comma while those of different categories cannot, etc.);
- combination of our strategies with strategies for the recognition of certain semantic categories (e.g., of city and region names, of human properties, etc.)

The middle-range goal of our project is to compile a lexicon for NLP that contains besides the standard lexical and semantic information functional information.

8. References

R. Dixon. 1982. *Where Have All the Adjectives Gone? and Other Essays in Semantics and Syntax*. Mouton, Berlin/Amsterdam/New York.

R. Dixon. 1991. *A New Approach to English Grammar, On Semantic Principles*. Clarendon Paperbacks, Oxford.

U. Engel. 1988. *Deutsche Grammatik*. Julius Groos Verlag, Heidelberg.

W. Frawley. 1992. *Linguistic Semantics*. Erlbaum, Hillsdale, NJ.

M.A.K. Halliday. 1994. *An Introduction to Functional Grammar*. Edward Arnold, London.

V. Hatzivassiloglou and K.R. McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the ACL '93*, pages 172–182, Ohio State University.

V. Hatzivassiloglou and K.R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the ACL '97*, pages 174–181, Madrid.

G. Helbig and J. Buscha. 1999. *Deutsche Grammatik. Ein Handbuch für den Ausländerunterricht*. Langenscheidt Verlag Enzyklopädie, Leipzig.

Stefan Klatt. forthcoming. *Ein Werkzeug zur Annotation von Textkorpora und Informationsextraktion*. Ph.D. thesis, Universität Stuttgart.

R. Quirk, S. Greenbaum, G. Leach, and J. Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, London.

V. Raskin and S. Nirenburg. 1995. Lexical semantics of adjectives. a microtheory of adjectival meaning. Technical Report MCCS-95-287, Computing Research Laboratory, New Mexico State University, Las Cruces, NM.

H. Seiler. 1978. Determination: A functional dimension for interlanguage comparison. In H. Seiler, editor, *Language Universals*. Narr, Tübingen.

J. Shaw and V. Hatzivassiloglou. 1999. Ordering among premodifiers. In *Proceedings of the ACL '99*, pages 135–143, University of Maryland, College Park.

G.H. Tucker. 1995. *The Treatment of Lexis in a Systemic Functional Model of English with Special Reference to Adjectives and their Structure*. Ph.D. thesis, University of Wales College of Cardiff, Cardiff.

Z. Vendler. 1968. *Adjectives and Nominalization*. Mouton, The Hague.

Word-level Alignment for Multilingual Resource Acquisition

Adam Lopez*, **Michael Nossal***, **Rebecca Hwa***, **Philip Resnik***†

*University of Maryland Institute for Advanced Computer Studies
†University of Maryland Department of Linguistics
College Park, MD 20742
{alopez, nossal, hwa, resnik}@umiacs.umd.edu

Abstract

We present a simple, one-pass word alignment algorithm for parallel text. Our algorithm utilizes synchronous parsing and takes advantage of existing syntactic annotations. In our experiments the performance of this model is comparable to more complicated iterative methods. We discuss the challenges and potential benefits of using this model to train syntactic parsers for new languages.

1 Introduction

Word alignment is an exercise commonly assigned to students learning a foreign language. Given a pair of sentences that are translations of each other, the students are asked to draw lines between words that mean the same thing.

In the context of multi-lingual natural language processing, word alignment (more simply, *alignment*) is also a necessary step for many applications. For instance, it is required in the parameter estimation step for training statistical translation models (Al-Onaizan et al., 1999; Brown et al., 1990; Melamed, 2000). Alignments are also useful for foreign language resource acquisition. Yarowsky and Ngai (2001) use an alignment to project part-of-speech (POS) tags from English to Chinese, and use the resulting noisy corpus to train a reliable Chinese POS tagger. Their result suggests that is worthwhile to consider more ambitious endeavors in resource acquisition.

Creating a syntactic treebank (e.g., the Penn Treebank Project (Marcus et al., 1993)) is time-consuming and expensive. As a consequence, state-of-the-art stochastic parsers which rely on such treebanks exist only in languages such as English for which they are available. If syntactic annotation could be projected from English to a language for which no treebank has been developed, then the treebank bottleneck may be overcome (Cabezas et al., 2001).

In principle, the success of treebank acquisition in this manner depends on a few key assumptions. The first assumption is that syntactic relationships in one language can be directly projected to another language using an accurate alignment. This theory is explored in Hwa et al. (2002b). A second assumption is that we have access to a reliable English parser and a word aligner. Although high-quality English parsers are available, high-quality aligners are more difficult to come by. Most alignment research has out of necessity concentrated on unsupervised methods. Even the best results are much worse than alignments created by humans. Therefore, this paper focuses on producing alignments that are tailored to the aims of syntactic projection. In particular, we propose a novel alignment model that, given an English sentence, its dependency parse tree, and its translation, simultaneously generates alignments and a dependency tree for the translation.

Our alignment model aims to improve alignment accuracy while maintaining sensitivity to constraints imposed by the syntactic transfer task. We hypothesize that the incorporation of syntactic knowledge into the alignment model will result in higher quality alignments. Moreover, by generating alignments and parse trees simultaneously, the alignment algorithm avoids irreconcilable errors in the projected trees such as crossing dependencies. Thus, our two objectives complement each other.

To verify these hypotheses, we have performed a suite of experiments, evaluating our algorithm on the quality of the resulting alignments and projected parse trees for English and Chinese sentence pairs. Our initial experiments demonstrate that our approach produces alignments and dependency trees whose quality is comparable to those produced by current state-of-the art systems.

We acknowledge that the strong assumptions we have stated for the success of treebank acquisition do not always hold true (Hwa et al., 2002a; Hwa et al., 2002b). Therefore, it will be necessary to devise a training algorithm that learns syntax even in the face of substantial noise introduced by failures in these assumptions. Although this last point is beyond the scope of this paper, we will allude to potential syntactic transfer approaches that are possible with our system, but infeasible under other approaches.

2 Background

Synchronous parsing appears to be the best model for syntactic projection. Synchronous parsing models the translation process as dual sentence generation in which a word and its translation in the other sentence are generated in lockstep. Translation pairs of both words and phrases are generated in a manner consistent with the syntax of their respective languages, but in a way that expresses the same relationship to the rest of the sentence. Thus, alignment and syntax are produced simultaneously and induce mutual constraints on each other. This model is ideal for the pursuit of our objectives, because it captures our complementary goals in an elegant theoretical framework.

Synchronous parsing requires both parses to adhere to the constraints of a given monolingual parsing model. If we assume context-free grammars, then each parse must be context-free. If we assume dependency grammars, then each parse must observe the planarity and connectivity constraints typical of such grammars (e.g. Sleator and Temperley (1993)).

In contrast, many alignment models (Melamed, 2000; Brown et al., 1990) rely on a bag-of-words model. This model presupposes no structural constraints on either input sentence beyond its linear order. To see why this type of model is problematic for syntactic transfer, consider what happens when syntax subsequently interacts with its output. Projecting dependencies across such an alignment may result in a dependency tree that violates planarity and connectivity constraints (Figure 1).

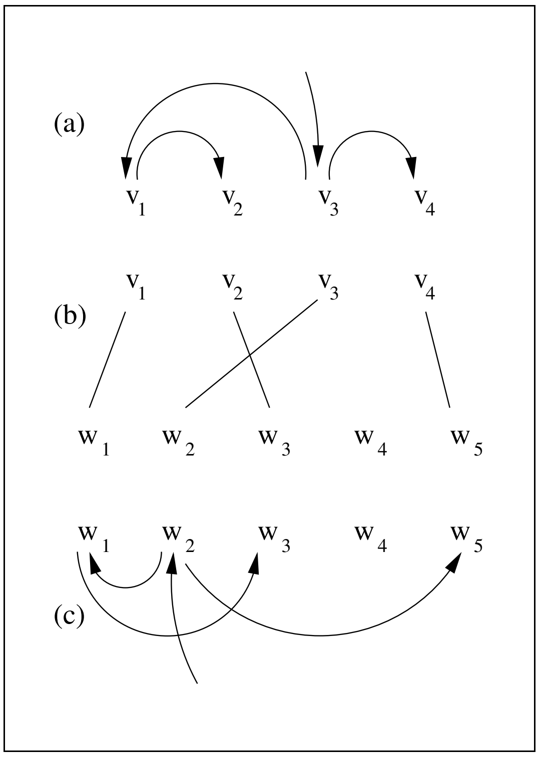


Figure 1: Violation of dependency grammar constraints caused by projecting a dependency parse across a bag-of-words alignment. Combining the syntax of (a) with the alignment of (b) produces the syntax of (c). In this example, the link (w_1, w_3) crosses the link (w_2, w_5) violating the planarity constraint. The word w_4 is unconnected, violating the connectivity constraint.

Once the fundamental assumptions of the syntactic model have been breached, there is no clear way to recover. For this reason, we would prefer not to use bag-of-words alignment models, although in many respects they remain state-of-the-art for alignment.

A canonical example of synchronous parsing is the Stochastic Inversion Transduction Grammar (SITV) (Wu, 1995). The SITV model imposes the constraints of context-free grammars on the synchronous parsing environment. However, we regard context-free grammars as problematic for our task, because recent statistical parsing models (Charniak, 2000; Collins, 1999; Ratnaparkhi, 1999) owe much of their success to ideas inherent to dependency parsing. We therefore adopt an algorithm described in Alshawi and Douglas (2000).[1] Their algorithm constructs synchronous dependency parses in the context of a domain-specific speech-to-speech translation system. In their system, synchronous parsing only enforces a contiguity constraint on phrasal translations. The actual syntax of the sentence is not assumed to be known. Nevertheless, their model is a synchronous parser for dependency syntax, and we adopt it for our purposes.

3 Our Modified Alignment Algorithm

We introduce parse trees as an optional input to the algorithm of Alshawi and Douglas (2000). We require that output dependency trees conform to dependency trees that are provided as input. If no parse tree is provided, our algorithm behaves identically to that of Alshawi and Douglas (2000).

3.1 Definitions

Our input is a parallel corpus that has been segmented into sentence pairs. We represent a sentence pair as the pair of word sequences ($V = v_1...v_m$, $W = w_1...w_n$). The algorithm iterates over the sentence pairs producing alignments.

We define a dependency parse as a rooted tree in which all words of the sentence appear once, and each node in the tree is such a word (Figure 2). An in-order traversal of the tree produces the sentence. A word is said to be *modified* by any words that appear as its children in the tree; conversely, the parent of a word is known as its *headword*. A word is said to *dominate* the span of all words that are descended from it in the tree, and is likewise known as the *headword* of that span.[2] Subject to these constraints, the dependency parse of V is expressed as a function $p_V : \{1...m\} \rightarrow \{0...m\}$ which defines the headword of each word in the dependency graph. The expression $p_V(i) = 0$ indicates that word v_i is the root node of the graph (the headword of the sentence). The dependency parse of W, $p_W : \{1...n\} \rightarrow \{0...n\}$ is defined in the same way.

An alignment is expressed as a function $a : \{1...m\} \rightarrow \{0...n\}$ in which $a(i) = j$ indicates that word v_i of V is aligned with word w_j of W. The case in which $a(i) = 0$ denotes *null alignment* (i.e. the word v_i does not correspond to any word in W). Under the constraints of synchronous parsing, we require that if $a(i) \neq 0$, then $p_W(a(i)) = a(p_V(i))$. In other words, the headword of a word's translation is the translation of the word's headword (Figure 3). We also require that the analogous condition hold for the inverse alignment map $a^{-1} : \{1...n\} \rightarrow \{0...m\}$.

3.2 Algorithm Details

Our algorithm (Appendix) is a bottom-up dynamic programming procedure. It is initialized by considering all

[1]An alternative to dependency grammar is the richer formalism of Synchronized Tree-Adjoining Grammar (TAG) (Shieber and Schabes, 1990). However, Synchronized TAG raises issues of computational complexity and has not yet been exploited in a stochastic setting.

[2]Elsewhere, the terms *connectivity* and *planarity* are used to define these constraints.

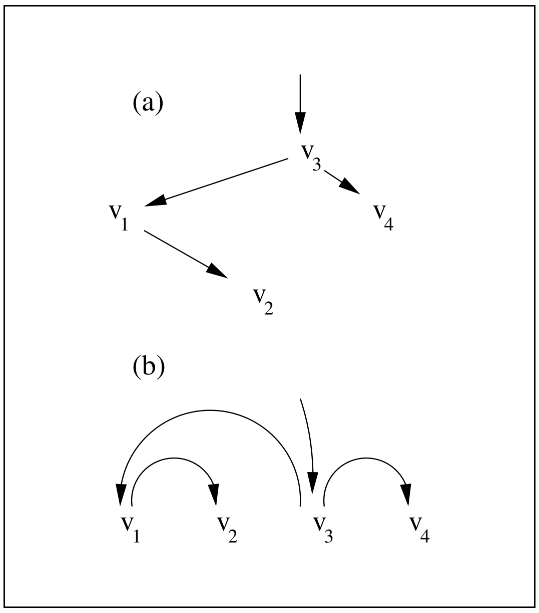


Figure 2: A dependency parse. In (a) the sentence is depicted in a tree form that makes the dominance and headword relationships clear (v_3 is the headword of the sentence). In (b) the same tree is depicted in more familiar sentence form, with the links drawn above the words.

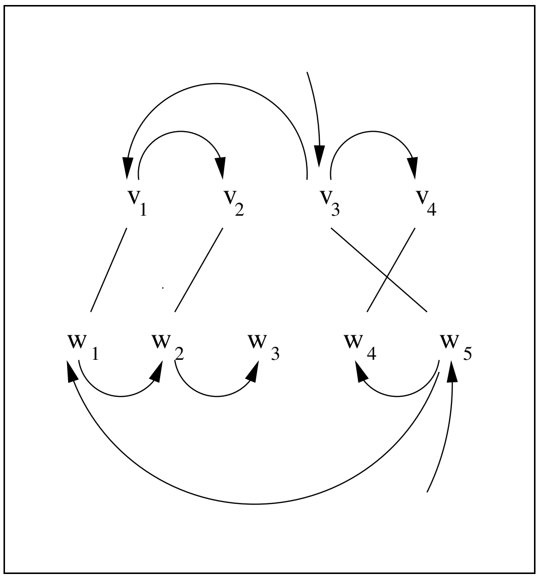


Figure 3: Synchronous dependency parses. Notice that all dependency links are symmetric across the alignment. In addition, the unaligned word w_3 is connected in the parse of W.

possible alignments of one word to another word or to null. Alshawi and Douglas (2000) considered alignments of two words to one or no words, but we found in our evaluations that restricting the initialization step to one word produced better results. In fact, Melamed (2000) argues in favor of exclusively one-to-one alignments. However, we may later explore in more detail the effects of initializing from multiword alignments.

As in Alshawi and Douglas (2000) each possible one-to-one alignment is scored using the ϕ^2 metric (Gale and Church., 1991), which is used to compute the correlation between $v_i \in V$ and $w_j \in W$ over all sentence pairs (V, W) in the corpus. Sentence co-occurrence counts are not the only possible data set with which we can use this metric. Therefore, we denote this type of initialization by ϕ^2_A to distinguish from a case we consider in Section 4.7, in which we use ϕ^2 initialized from counts of Giza++ alignment links. The latter case is denoted by ϕ^2_G.

To compute alignments of larger spans, the algorithm combines adjacent sub-alignments. During this step, one sub-alignment becomes a modifier phrase. Interpreting this in terms of dependency parsing, the aligned headwords of the modifier phrase become modifiers of the aligned headwords of the other phrase. At each step, the score of the alignment is computed. Following Alshawi and Douglas (2000) we simply add the score of the sub-alignments. Thus the overall score of any aligned subphrase can be computed as follows.

$$\sum_{(i,j):a(i)=j} \phi^2(v_i, w_j)$$

The output of the algorithm is simply the highest-scoring alignment that covers the entire span of both V and W.

3.3 Treatment of Null Alignments

Null alignments present a few practical issues. For experiments involving ϕ^2_A, we adopt the practice of counting a null token in the shorter sentence of each pair.[3] An alternative solution to this problem would involve initialization from a word association model that explicitly handles nulls, such as that of Melamed (2000).

An implication of the synchronous parsing constraint given in Section 3.1 is that null aligned words must be leaf words within their respective dependency graphs. In certain cases this may not lead to the best synchronized parse. We remove this condition. Effectively, we consider each sentence to consist of the same number of tokens, some of which may be null tokens. (usually, this will introduce null tokens into only the shorter sentence, but not necessarily). The null tokens behave like words with regards to the synchronous parsing constraint, but they do not impact phrase contiguity.[4] In only the resulting surface dependency graphs, we remove null tokens by contracting all edges between the null token and its parent and naming the resultant node with the word on the parent node. Recall from graph theory that contraction is an operation whereby an edge is removed and the nodes at its endpoints are conflated. [5] Thus, words that modify a null token are interpreted as modifiers of the the null token's headword. This is illustrated in Figure 4. One important implication of this is that we can only allow a null token to be the headword

[3]Srinivas Bangalore, personal communication.

[4]a null token is considered to be contiguous with any other subphrase – another way to view this is that a null token is an unseen word that may appear at any location in the sentence in order to satisfy contiguity constraints.

[5]see e.g., Gross and Yellen (1999)

of the sentence if it has a single modifier. Otherwise, the result of the graph contraction would not be a rooted tree. We found that this treatment of null alignments resulted in a slight improvement in alignment results.

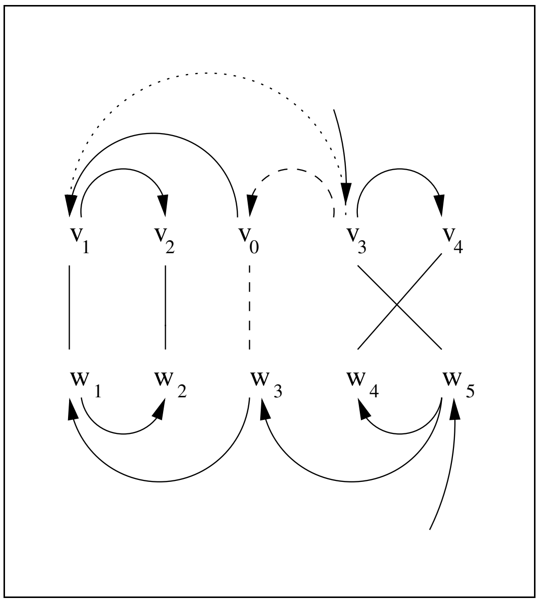


Figure 4: Effect of null words on synchronous parses. In this case, word w_3 has been aligned to the null token v_0. However, v_0 can still dominate other words in the parse of V. Once the structure has been completed, the edge between v_0 and v_3 (indicated by the dashed line) will contract. This will cause the dependency between v_1 and v_0 to become the inferred dependency (indicated by the dotted line) between v_1 and v_3.

3.4 Analysis

In the case that there are no parses available, the computational complexity of the algorithm is $O(m^3n^3)$, but with a parse of V (and an efficient enumeration of the subphrase combinations allowed by the parse) the complexity reduces to $O(m^3n)$. If both parses are available the complexity would be reduced to $O(mn)$.

It is important to note that as it is presented, our algorithm does not search the entire space of possible alignment/tree combinations. Melamed observes that two modifications are required to accomplish this.[6] The first modification entails the addition of four new loop parameters to enumerate the possible headwords of the four monolingual subspans. These additional parameters add a factor of $O(m^2n^2)$. Second, Melamed points out that for a small subset of legal structures, it must be possible to combine subphrases that are not adjacent to one another. The most efficient solution to this problem adds two more parameters, for a total of $O(m^6n^6)$. The best known optimization reduces the total complexity to $O(m^5n^5)$. This is far too complex for a practical implementation, so we chose to use the original $O(m^3n^3)$ algorithm for our evaluations. Thus we recognize that our algorithm does not search the entire space of synchronous parses. It inherently incorporates a greedy heuristic, since for each subphrase, it considers only the most likely headword.

[6]I. Dan Melamed, personal communication.

4 Evaluation

We have performed a suite of experiments to evaluate our alignment algorithm. The qualities of the resulting alignments and dependency parse trees are quantified by comparisons with correct human-annotated parses. We compare the alignment output of our algorithm with that of the basic algorithm described in Alshawi and Douglas (2000) and the well-known IBM statistical model described in Brown et al. (1990) using the freely available implementation (Giza++) described in Al-Onaizan et al. (1999). We also compare the output dependency trees against several baselines and against projected dependency trees created in the manner described in (Hwa et al., 2002a). We found that our model, which combines cross-lingual statistics with syntactic annotation, produces alignments and trees that are are comparable to the best results of other methods.

4.1 Data Set

The language pair we have focused on for this study is English-Chinese. The training corpus consists of around 56,000 sentence pairs from the Hong Kong News parallel corpus. Because the training corpus is solely used for word co-occurrence statistics, no annotation is performed on it.

The development set was constructed by obtaining manual English translations for 47 Chinese sentences of 25 words or less, taken from sections 001-015 of the Chinese Treebank (Xia et al., 2000). A separate test set, consisting of 46 Chinese sentences of 25 words or less, was constructed in a similar fashion.[7] To obtain correct English parses, we used a context-free parser (Collins, 1999) and converted its output to dependency format. To obtain correct Chinese parses, Chinese Treebank trees were converted to dependency format. Both sets of parses were hand-corrected. The correct alignments for the development and test set were created by two native Chinese speakers using annotation software similar to that described in Melamed (1998).

4.2 Metrics for evaluating alignments

As a measure of alignment accuracy, we report Alignment Precision (AP) and Alignment Recall (AR) figures. These are computed by by comparing the alignment links made by the system with the links in the correct alignment. We denote the set of guessed alignment links by G_a and the set of correct alignment links by C_a. Precision is given by $AP = \frac{|C_a \cap G_a|}{|G_a|}$. Recall is given by $AR = \frac{|C_a \cap G_a|}{|C_a|}$. We also compute the F-score (AF), which is given by $AF = \frac{2 \cdot AP \cdot AR}{AP + AR}$. Null alignments are ignored in all computations. Our evaluation metric is similar to that of Och and Ney (2000).

[7]These sentences have already been manually translated into English as part of the NIST MT evaluation preview (See http://www.nist.gov/speech/tests/mt/). The sentences were taken from sections 038, 039, 067, 122, 191, 207, 249.

Synchronous Parsing Method	AP	AR	AF	CTP
sim-Alshawi (ϕ^2_A)	40.6	36.5	38.4	18.5
sim-Alshawi (ϕ^2_A) + English parse	43.8	39.3	41.4	39.9
sim-Alshawi (ϕ^2_A) + English parse + Chinese bigrams	42.9	38.5	40.6	39.4
sim-Alshawi (ϕ^2_A) + both bigrams	41.5	37.3	39.3	16.5
Giza++ initialization (ϕ^2_G)	51.2	45.9	**48.4**	11.6
Giza++ initialization (ϕ^2_G)+ English parse	49.6	44.6	47.0	**44.7**

Baseline Method	AP	AR	AF	CTP
Same Order Alignment	15.7	14.1	14.8	NA
Random Alignment (avg scores)	7.8	7.0	7.4	NA
Forward-chain	NA	NA	NA	37.3
Backward-chain	NA	NA	NA	12.9
Giza++	68.7	40.9	**51.3**	NA
Hwa et al. (2002a)	NA	NA	NA	**44.1**

Table 1: Alignment Results for All Methods.
AP = Alignment Precision. AR = Alignment Recall. AF = Alignment F-Score. CTP = Chinese Tree Precision.
All scores are reported as percentages of 100.
The best scores in each table appear in bold.

4.3 Metrics for evaluating projected parse trees

As a measure of induced dependency tree accuracy, we report unlabeled Chinese Tree Precision (CTP). This is computed by comparing the output dependency tree with the correct dependency trees. We denote the set of guessed dependency links by G_p and the set of correct alignment links by C_p. A small number of words (mostly punctuation) were not linked to any parent word in the correct parse; links containing these words are not included in either C_p or G_p. Precision is given by $CTP = \frac{|C_p \cap G_p|}{|G_p|}$. For dependency trees, $|C_p| = |G_p|$, since each word contributes one link relating it to its headword. Thus, recall is the same as precision for our purposes.

4.4 Baseline Results

We first present the scores of some naïve algorithms as a baseline in order to provide a lower bound for our results. The results of the baseline experiments are included with all other results in Table 1. Our first baseline (Same Order Alignment) simply maps character v_i in the English sentence to character w_i in the Chinese sentence, or w_n in the case of $i > n$. Our second baseline (Random Alignment), randomly aligns word v_i to word w_j subject to the constraint that no words are multiply aligned. We report the average scores over 100 runs of this baseline. The best Random Alignment F-score was 10.0% and the worst was 5.3% with a standard deviation of 0.9%.

For parse trees, we use two simple baselines. In the first (Forward-Chain), each word modifies the word immediately following it, and the last word is the headword of the sentence. For the second baseline (Backward-Chain), each word modifies the word immediately preceding it, and the first word is the headword of the sentence. No alignment was performed for these baselines.

The remaining baselines relate to the Giza++ algorithm. Giza++ produces the best word alignments. For reasons described previously, Giza++ alignments do not combine easily with syntax. However, Hwa et al. (2002a) contains an investigation in which trees output from a projection across Giza++ alignment are modified using several heuristics, and subsequently improved using linguistic knowledge of Chinese. We report the Chinese Tree Precision obtained by this method.

4.5 Synchronous Parsing Results

Our first set of alignments combines the ϕ^2_A cross-lingual co-occurrence metric described previously with either English parse or no parse trees. In this set, ϕ^2_A with no parse is nearly identical to the approach described in Alshawi and Douglas (2000) (excepting our treatment of null alignments). Thus, it serves as a useful point of comparison for runs that make use of other information. In Table 1 we refer to it as sim-Alshawi.

What we find is that incorporating parse trees results in a modest improvement over the baseline approach of sim-Alshawi. Why aren't the improvements more substantial? One observation is that using parses in this manner results in only passive interaction with the cross-lingual ϕ^2_A scores. In other words, the parse filters out certain alignments, but cannot in any other way counteract the biases inherent in the word statistics. Nevertheless, it appears to be modest progress.

4.6 Results of Using Bigrams to Approximate Parses

The results suggest that using parses to constrain the alignment is helpful. It is possible that using both parses would result in a more substantial improvement. However, we have already stated that we are interested in the case of asynchronous resources. Under this scenario, we only have access to one parse. Is there some way that we can approximate syntactic constraints of a sentence without having access to its parse?

The parsers of (Charniak, 2000; Collins, 1999; Ratnaparkhi, 1999) make substantial use of *bilexical dependencies*. Bilexical dependencies capture the idea that linked words in a dependency parse have a statistical affinity for each other: they often appear together in certain contexts. We suspect that bigram statistics could be used as a proxy for actual bilexical dependencies.

We constructed a simple test of this theory: for each English sentence $V = v_1...v_m$ in the development set with parse $p_V : \{1...m\} \rightarrow \{0...m\}$, we first construct the set of all bigrams $B = \{(v_i, v_j) : 1 \leq i < j \leq m\}$. We then partitioned B into two sets: bigrams of linked words, i.e. $L = \{(v_i, v_j) : (v_i, v_j) \in B; p_V(v_i) = v_j \text{ or } p_V(v_j) = v_i\}$ and unlinked words $U = B - L$. We used the Bigram Statistics Package (Pedersen, 2001), to collect bigram statistics over the entire dev/train corpus and compute the average statistical correlation of each set using a variety of metrics (loglikelihood, dice, χ^2, ϕ^2). The results indicated that bigrams in the linked set L were more correlated than those in the unlinked set U under all metrics. We repeated this experiment with the development sentences in Chinese, with similar results. Although this is by no means a conclusive experiment, we took the results as an indication that using bigram statistics as an approximation of a parse might be helpful where no parse was actually available.

To incorporate bigram statistics into our alignment model, we modified the scoring function in the following manner: each time a dependency link is introduced between words and we do not have access to the source parse, we add into the alignment score the bigram score of the two words. The bigram score is based on the ϕ^2 metric computed for bigram correlation. We call this ϕ_B^2. The resulting alignment score can now be given by the following formula.

$$\sum_{(i,j):a(i)=j} \phi_A^2(v_i, w_j) + \sum_{(i,j):i<j, p_W(i)=j \wedge p_W(j)=i} \phi_B^2(w_i, w_j)$$

Our results indicate that using Chinese bigram statistics in conjunction with English parse trees in this manner results in a small decrease in the score along all measures. Nonetheless, there is an intuitively appealing interpretation of using bigrams in this way. The first is that the modification of the scoring function provides competitive interaction between parse information and cross-lingual statistics. The second is that if bigram statistics represent a weak approximation of syntax, then perhaps the iterative refinement of this statistic (e.g. by taking counts only over words that were linked in a previous iteration) would satisfy our objective of syntactic transfer.

4.7 Results of Using Better Word Statistics

Our results show that using parse information and coarse cross-lingual word statistics provides a modest boost over an approach using only the cross-lingual word statistics. We also decided to investigate what happens when we seed our algorithm with better cross-lingual statistics

To test this, we initialize our co-occurrence counts from alignment links output by the Giza++ alignment of our corpus. We still use ϕ^2 to compute the correlation. We call this ϕ_G^2. Predictably, using the better word correlation statistics improves the quality of the alignment output in all cases. In this scenario, adding parse information does not seem to improve the alignment score. However, parse trees induced in this manner achieve a higher precision than any of the other methods. It outscores the baseline algorithms by a significant amount, and produces results comparable to the baseline of Hwa et al. (2002a). It is important to note, however, that the baseline of Hwa et al. (2002a) is achieved only after the application of numerous linguistic rules to the output of the Giza++ alignment. Additionally, the trees themselves may contain errors of the type described in Section 2. In contrast, our tree precision results directly from the application of our synchronous parsing algorithm, and all of the output trees are valid dependency parses.

5 Future Work

We believe that a fundamental advantage of our baseline model is its simplicity. Improving upon it will be considerably easier than improving upon a complex model such as the one described in Brown et al. (1990). Improvements may proceed along several possible paths. One path would involve reformulating the scoring functions in terms of statistical models (e.g. generative models). A natural complement to this path would be the introduction of iteration with the goal of improving the alignments and the accompanying models. In this approach, we could attempt to learn a coarse statistical model of the syntax of the low-density language after each iteration of the alignment. This information could in turn be used as evidence in the next iteration of the alignment model, hopefully improving its performance. Our results have already established a set of statistics that could be used in the initial iteration of such a task. The iterative approach resonates with an idea proposed in Yarowsky and Ngai (2001), regarding the use of learned part-of-speech taggers in subsequent alignment iterations.

An orthogonal approach would be the application of additional linguistic information. Our results indicated that syntactic knowledge can help improve alignment. Additional linguistic knowledge obtained from named-entity analyses, phrasal boundary detection, and part-of-speech tags might also improve alignment.

Although our output dependency trees represent definite progress, trees with such low precision cannot be used directly to train statistical parsers that assume correct training data (Charniak, 2000; Collins, 1999; Ratnaparkhi, 1999). There are two possible methods of improving upon the precision of this training data. The first is the use of noise-resistant training algorithms such as those described in (Yarowsky and Ngai, 2001). The second is the possibility of improving the precision yield by removing obviously bad training examples from the set. Unlike the baseline model, our word alignment model provides an obvious means of doing this. One possibility is to use a score gleaned from the alignment algorithm as a means of ranking dependency links, and removing links whose score is above some threshold. We hope that a dual approach of improving the precision of the training examples, while simultaneously reducing the sensitivity of the training algorithm, will result in the ability to train a reasonably accurate statistical parser for the new language. Our eventual objective is to train a parser in this manner.

6 Related work

Al-Onaizan et al. (1999), Brown et al. (1990) and Melamed (2000) focus on the description of statistical translation models based on the bag-of-words model. Alignment plays a crucial part in the parameter estimation methods of these models, but they remain problematic for syntactic transfer for reasons described in Section 2. The work of Hwa et al. (2002b) is an investigation into the combination of syntax with the output of this type of model. Och et al. (1999) presents a statistical translation model that performs phrasal translation, but it relies on shallow phrases that are discovered statistically, and makes no use of syntax. Yamada and Knight (2001) create a full-fledged syntax-based translation model. However, their model is unidirectional; it only describes the syntax of one sentence, and makes no provision for the syntax of the other. Wu (1995) presents a complete theory of synchronous parsing using a variant of context-free grammars, and exhibits several positive results, though not for syntax transfer. Alshawi and Douglas (2000) present the synchronous parsing algorithm on which our work is based. Much like the work on translation models, however, this work is interested in alignment primarily as a mechanism for training a machine translation system. Variations on the synchronous parsing algorithm appear in Alshawi et al. (2000a) and Alshawi et al. (2000b), but the algorithm of Alshawi and Douglas (2000) appears to be the most complete.

7 Conclusion

We have described a new approach to alignment that incorporates dependency parses into a synchronous parsing model. Our results indicate that this approach results in alignments whose quality is comparable to those produced by complicated iterative techniques. In addition, our approach demonstrates substantial promise in the task of learning syntactic models for resource-poor languages.

8 Acknowledgements

This work has been supported, in part, by ONR MURI Contract FCPO.810548265, DARPA/ITO Cooperative Agreement N660010028910, NSA Contract RD-02-5700 and Mitre Contract 010418-7712. The authors would like to thank I. Dan Melamed and Srinivas Bangalore for helpful discussions; Franz Josef Och for help with Giza++; and Lingling Zhang, Edward Hung, and Gina Levow for creating the gold standard annotations for the development and test data.

9 References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, I. Dan Melamed, Franz Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation: Final report. In *Summer Workshop on Language Engineering*. John Hopkins University Center for Language and Speech Processing.

Hiyan Alshawi and Shona Douglas. 2000. Learning dependency transduction models from unannotated examples. *Philosophical Transactions of the Royal Society*, 358:1357–1372.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000a. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26:1357–1372.

Hiyan Alshawi, Srinivasa Bangalore, and Shona Douglas. 2000b. Head transducer models for speech translation and their automatic acquisition from bilingual data. *Machine Translation*, 15:105–124.

Peter F. Brown, John Cocke, Stephen Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Clara Cabezas, Bonnie Dorr, and Philip Resnik. 2001. Spanish language processing at university of maryland: Building infrastructure for multilingual applications. In *Proceedings of the Second International Workshop on Spanish Language Processing and Language Technologies (SLPLT-2)*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

William A. Gale and Kenneth W. Church. 1991. Identifying word correspondences in parallel texts. In *Proceedings of the Fourth DARPA Speech and Natural Language Processing Workshop*, pages 152–157.

Jonathan Gross and Jay Yellen, 1999. *Graph Theory and Its Applications*, chapter 7.5: Transforming a Graph by Edge Contraction, pages 263–266. Series on Discrete Mathematics and Its Applications. CRC Press.

Rebecca Hwa, Philip Resnik, and Amy Weinberg. 2002a. Breaking the resource bottleneck for multilingual parsing. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*. To appear.

Rebecca Hwa, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002b. Evaluating translational correspondence using annotation projection. In *Proceedings of the 40th Annual Meeting of the ACL*. To appear.

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

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

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, Jun.

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

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, Jun.

Ted Pedersen. 2001. A decision tree of bigrams is an accurate predictor of word sense. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 79–86, Jun.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Stuart Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, volume 3, pages 1–6.

Daniel Sleator and Davy Temperley. 1993. Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*, Aug.

Dekai Wu. 1995. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1328–1335, Aug.

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

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the Conference of the Association for Computational Linguistics*.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, Jun.

A Algorithm Pseudocode

The following code does not address what constitutes a legal combination of subspans for an alignment. Legal subspans depend on constraints imposed by an input parse, if available. Otherwise, as in Alshawi and Douglas (2000), all possible combinations of subspans are legal. Regardless of what constitutes a legal subspan, the enumeration of spans must be done in a reasonable way. Small spans must be enumerated before larger spans that are constructed from them.

The variables i_V and j_V denote the span $v_{i_V+1}...v_{j_V}$, and p_V denotes a partition of the span such that $i_V \leq p_V \leq j_V$. The variables i_W, j_W, and p_W are defined analogously on W.

Our data structure is a chart α, which contains cells indexed by i_V, j_V, i_W, and j_W. Each cell contains subfields $phrase$, $modifierPhrase$, and $score$.

Finally, we assume the existence of functions assocScore and score. The assocScore function computes the score of directly aligning to short spans of the sentence pair. In this paper, we use variations on the ϕ^2 metric (Gale and Church., 1991) for this. The score function computes the score of combining two sub-alignments, assuming that the second sub-alignment becomes a modifier of the first. In this paper, we use one score function that simply adds the score of sub-alignments, and one that adds bigram correlation to the score of the sub-alignments. In principle, arbitrary scoring functions can be used.

```
initialize the chart
for all legal combinations of $i_V$, $j_V$,$i_W$, and $j_W$
    $\alpha(i_V, j_V, i_W, j_W) = assocScore(v_{i_V+1}...v_{j_V}, w_{i_W+1}...w_{j_W})$
complete the chart
for all legal combinations of $i_V$, $j_V$, $p_V$, $i_W$, $j_W$, and $p_W$
    consider the case in which aligned subphrases are in the same order in both languages.
    $phrase = \alpha(i_V, p_V, i_W, p_W)$
    $modifierPhrase = \alpha(p_V, j_V, p_W, j_W)$
    $score =$score($phrase, modifierPhrase$)
    if $score > \alpha(i_V, j_V, i_W, j_W).score$ then
        $\alpha(i_V, j_V, i_W, j_W) =$ new subAlignment($phrase, modifierPhrase, score$)
    consider the case in which the dominance relationship between these two phrases is reversed.
    swap($phrase, modifierPhrase$)
    $score =$score($phrase, modifierPhrase$)
    if $score > \alpha(i_V, j_V, i_W, j_W).score$ then
        $\alpha(i_V, j_V, i_W, j_W) =$ new subAlignment($phrase, modifierPhrase, score$)
    consider the case in which aligned subphrases are in the reverse order in each language.
    $phrase = \alpha(i_V, p_V, p_W, j_W)$
    $modifierPhrase = \alpha(p_V, j_V, i_W, p_W)$
    $cost =$cost($phrase, modifierPhrase$)
    $score =$score($phrase, modifierPhrase$)
    if $score > \alpha(i_V, j_V, i_W, j_W).score$ then
        $\alpha(i_V, j_V, i_W, j_W) =$ new subAlignment($phrase, modifierPhrase, score$)
    consider the case in which the dominance relationship between these two phrases is reversed.
    swap($phrase, modifierPhrase$)
    $score =$score($phrase, modifierPhrase$)
    if $score > \alpha(i_V, j_V, i_W, j_W).score$ then
        $\alpha(i_V, j_V, i_W, j_W) =$ new subAlignment($phrase, modifierPhrase, score$)
return $\alpha(0, m, 0, n)$
```

Generating A Parsing Lexicon from an LCS-Based Lexicon

Necip Fazıl Ayan and Bonnie J. Dorr

Department of Computer Science
University of Maryland
College Park, 20742, USA
{nfa, bonnie}@umiacs.umd.edu

Abstract

This paper describes a technique for generating parsing lexicons for a principle-based parser (Minipar). Our approach maps lexical entries in a large LCS-based repository of semantically classified verbs to their corresponding syntactic patterns. A by-product of this mapping is a lexicon that is directly usable in the Minipar system. We evaluate the accuracy and coverage of this lexicon using LDOCE syntactic codes as a gold standard. We show that this lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). In a later experiment, we automate the process of mapping between the LCS-based repository and syntactic patterns. The advantage of automating the process is that the same technique can be applied directly to lexicons we have for other languages, for example, Arabic, Chinese, and Spanish.

1. Introduction

This paper describes a technique for generating parsing lexicons for a principle-based parser (Minipar (Lin, 1993; Lin, 1998)) using a lexicon that is semantically organized according to Lexical-Conceptual Structure (LCS) (Dorr, 1993; Dorr, 2001)—an extended version of the verb classification system proposed by (Levin, 1993).[1] We aim to determine how much syntactic information we can obtain from this resource, which extends Levin's original classification as follows: (1) it contains 50% more verbs and twice as many verb entries (Dorr, 1997)—including new classes to accommodate previously unhandled verbs and phenomena (e.g., clausal complements); (2) it incorporates theta-roles which, in turn, are associated with a thematic hierarchy for generation (Habash and Dorr, 2001); and (3) it provides a higher degree of granularity, i.e., verb classes are sub-divided according to their aspectual characteristics (Olsen et al., 1997).

More specifically, we provide a general technique for projecting this broader-scale semantic (language-independent) lexicon onto syntactic entries, with the ultimate objective of testing the effects of such a lexicon on parser performance. Each verb in our semantic lexicon is associated with a class, an LCS representation, and a thematic grid.[2] These are mapped systematically into syntactic representations. A by-product of this mapping is a lexicon that is directly usable in the Minipar system.

Several recent lexical-acquisition approaches have produced new resources that are ultimately useful for syntactic analysis. The approach that is most relevant to ours is that of (Stevenson and Merlo, 2002b; Stevenson and Merlo, 2002a), which involves the derivation of verb classes from syntactic features in corpora. Because their approach is unsupervised, it provides the basis for automatic verb classification for languages not yet seen. This work is instrumental in providing the basis for wide-spread applicability of our technique (mapping verb classes to a syntactic parsing lexicon), as verb classifications become increasingly available for new languages over the next several years.

An earlier approach to lexical acquisition is that of (Grishman et al., 1994), an effort resulting in a large resource called Comlex—a repository containing 38K English headwords associated with detailed syntactic patterns. Other researchers (Briscoe and Carroll, 1997; Manning, 1993) have also produce subcategorization patterns from corpora. In each of these cases, data collection is achieved by means of statistical ex-

[1]We focus only on verb entries as they are cross-linguistically the most highly correlated with lexical-semantic divergences.

[2]Although Lexical Conceptual Structure (LCS) is the primary semantic representation used in our verb lexicon, it is not described in detail here (but see (Dorr, 1993; Dorr, 2001). For the purpose of this paper, we rely primarily on the thematic grid representation, which is derived from the LCS. Still we refer to the lexicon as "LCS-based" as we store all of these components together in one large repository: http://www.umiacs.umd.edu/~bonnie/LCS_Database_Documentation.html.

traction from corpora; there is no semantic basis and neither is intended to be used for multiple languages.

The approaches of (Carroll and Grover, 1989) and (Egedi and Martin, 1994) involve acquisition English lexicons from entries in LDOCE and *Oxford Advanced Learner's Dictionary* (OALD), respectively. The work of (Brent, 1993) produces a lexicon from a grammar—the reverse of what we aim to do. All of these approaches are specific to English. By contrast, our goal is to have a unified repository that is transferable to other languages—and from which our parsing (and ultimately generation) grammars may be derived.

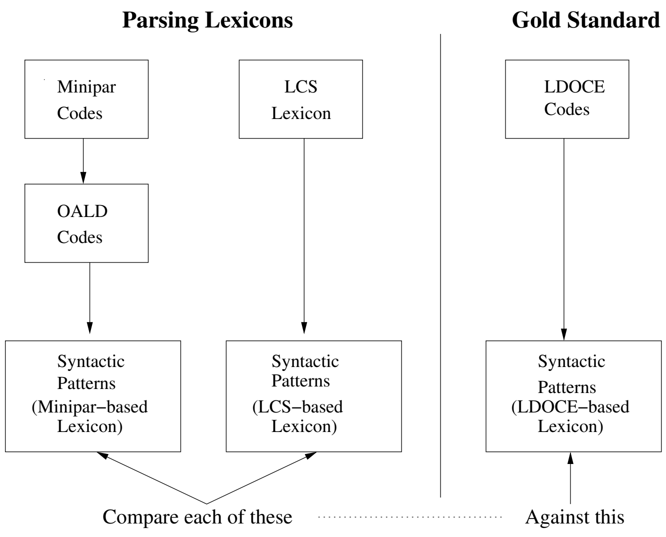
For evaluation purposes, we developed a mapping from the codes of Longman's Dictionary of Contemporary English (LDOCE (Procter, 1978))—the most comprehensive online dictionary for syntactic categorization—to a set of syntactic patterns. We use these patterns as our gold standard and show that our derived lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). In a later experiment, we automate the process of mapping between the LCS-based repository and syntactic patterns—with the goal of portability: We currently have LCS lexicons for English, Arabic, Spanish, and Chinese, so our automated approach allows us to produce syntactic lexicons for parsing in each of these languages.

Section 2. presents a brief description of each code set we use in our experiments. In Section 3., we explain how we generated syntactic patterns from three different lexicons. In Section 4., we discuss our experiments and the results. Section 5. describes ongoing work on automating the mapping between LCS-based representations and syntactic patterns. Finally, we discuss our results and some possible future directions.

2. Code Descriptions

In many online dictionaries, verbs are classified according to the arguments and modifiers that can follow them. Most dictionaries use specific codes to identify transitivity, intransitivity, and ditransitivity. These broad categories may be further refined, e.g., to distinguish verbs with NP arguments from those with clausal arguments. The degree of refinement varies widely.

In the following subsections, we will present three different code sets. As shown in Figure 1, the first of these (OALD) serves as a mediating representation in the mapping between Minipar codes and syntactic patterns. The LCS lexicon and LDOCE codes are mapped directly into syntactic patterns, without an intervening representation. The patterns resulting from the LDOCE are taken as the gold standard, serving as the basis of comparison between the Minipar- and LCS-based lexicons.



Parsing Lexicons
Gold Standard
Minipar Codes
LCS Lexicon
LDOCE Codes
OALD Codes
Syntactic Patterns (Minipar–based Lexicon)
Syntactic Patterns (LCS–based Lexicon)
Syntactic Patterns (LDOCE–based Lexicon)
Compare each of these
Against this

Figure 1: A Comparison between Minipar- and LCS-based Lexicons using LDOCE as the Gold Standard

2.1. OALD Codes

This code set is used in Oxford Advanced Learner's Dictionary, a.k.a OALD (Mitten, 1992). The verbs are categorized into 5 main groups: Intransitive verbs, transitive verbs, ditransitive verbs, complex transitive verbs, and linking verbs. Each code is of the form $Sa_1[.a_2]$ where S is the first letter of the verb categorization ($S \in \{I, T, D, C, L\}$ for the corresponding groups), and $a_1, a_2, \ldots$ are the argument types. If a code contains more than one argument, each argument is listed serially. Possible argument types are n for nouns, f for finite clauses (*that* clauses), g for "-ing" clauses, t for infinitive clauses, w for finite clauses beginning with "-wh", i for bare infinitive clauses, a for adjective phrases, p for prepositions and pr for prepositional phrases.

For example, Tn refers to the verbs followed by a noun ('She read the book'), $Tn.pr$ refers to the verbs followed by a noun and a prepositional phrase ('He opened the door with a latch'), and $Dn.n$ refers to the verbs followed by two nouns ('She taught the children French'). The number of codes in OALD code set is 32 and the codes are listed in Table 1.

OALD codes are simplistic in that they do not include modifiers. In addition, they also do not explicitly specify which prepositions can be used in the PPs.

2.2. Minipar Codes

The Minipar coding scheme is an adaptation of the OALD codes. Minipar extends OALD codes by pro-

Categorization	OALD Codes
Intransitive verbs	{I, Ip, Ipr, In/pr, It}
Transitive verbs	{Tn, Tn.pr, Tn.p, Tf, Tw, Tt, Tg, Tn.t, Tn.g, Tn.i}
Complex Transitive verbs	{Cn.a, Cn.n, Cn.n/a, Cn.t, Cn.g, Cn.i}
Ditransitive verbs	{Dn.n, Dn.pr, Dn.f, Dn.t, Dn.w, Dpr.f, Dpr.w, Dpr.t}
Linking verbs	{La, Ln}

Table 1: OALD Code Set: The Basis of Minipar Codes

viding a facility for specifying prepositions, but only 8 verbs are encoded with these prepositional codes in the official Minipar distribution. In these cases, the codes containing pr are refined to be $pr.prep$, where $prep$ is the head of the PP argument.[3] In addition, Minipar codes are refined in the following ways:

1. Optional arguments are allowed, e.g., $T[n].pr$ describes verbs followed by an optional noun and a PP. This is equivalent to the combination of the OALD codes $Tn.pr$ and Ipr.
2. Two or more codes may be combined, e.g., *Tfgt* describes verbs followed by a clause that is finite, infinitive, or gerundive ("-ing").
3. Prepositions may be specified in prepositional phrases. Some of the codes containing *pr* as an argument are converted into *pr.prep* in order to declare that the prepositional phrase can begin with only the specified preposition *prep*.

The set of Minipar codes contain 66 items. We will not list them here since they are very similar to the ones in Table 1, with the modifications described above.

2.3. LDOCE Codes

LDOCE has a more detailed code set than that of OALD (and hence Minipar). The codes include both arguments and modifiers. Moreover, prepositions are richly specified throughout the lexicon. The syntax of the codes is either *CN* or *CN-Prep*, where *C* corresponds to the verb sub-categorization (as in the generic OALD codes) and N is a number, which corresponds to different sets of arguments that can follow the verb. For example, T1-ON refers to verbs that are followed by a noun and a PP with the head *on*. The number of codes included in this set is 179. The meaning of each is described in Table 2.

3. Our Approach

Our goal is evaluate the accuracy and coverage of a parsing lexicon where each verb is classified according to the arguments it takes. We use syntactic patterns as the basis of the comparison between our parsing lexicon and the original lexicon used in Minipar.

Number	Arguments
1	one or more nouns
2	bare infinitive clause
3	infinitive clause
4	-ing form
5	-that clause
6	clauses with a wh- word
7	adjective
8	past participle
9	descriptive word or phrase

Table 2: LDOCE Number Description

Syntactic patterns simply list the type of the arguments one by one, including the subject. Formally, a syntactic pattern is $a_1, a_2, \ldots$ where a_i is an element of NP, AP, PP, FIN, INF, BARE, ING, WH, PREP, corresponding to noun phrases, adjective phrases, prepositional phrases, clauses beginning with "that", infinitive clauses, bare infinitive clauses, "-ing" clauses, "-wh" clauses and prepositions, respectively. Prepositional phrases may be made more specific by including the heads, which is done by PP.$prep$ where $prep$ is the head of the prepositional phrase. The first item in the syntactic pattern gives the type of the subject.

Our initial attempts at comparing the Minipar- and LCS-based lexicons involved the use of the OALD code set instead of syntactic patterns. This approach has two problems, which are closely related. First, using the class number and thematic grids as the basis of mapping from the LCS lexicon to OALD codes is a difficult task because of the high degree of ambiguity. For example, it is hard to choose among four OALD codes (Ln, La, Tn or Ia) for the thematic grid _th_pred, regardless of the Levin class. In general, the grid-to-OALD mapping is so ambiguous that maintaining consistency over the whole LCS lexicon is virtually impossible.

Secondly, even if we are able to find the correct OALD codes, it is not worth the effort because all that is needed for the parsing lexicon is the type and number of arguments that can follow the verb. For example, $Cn.n$ (as in *"appoint him king"*) and $Dn.n$ (as in *"give him a book"*) both correspond to two

[3]This extension is used only for the preposition *as* for the verbs *absolve, accept, acclaim, brand, designate, disguise, fancy,* and *reckon*.

NPs, but the second NP is a direct object in the former case and an indirect object in the latter. Since the parser relies ultimately on syntactic patterns, not codes, we can eliminate this redundancy by mapping any verb in either of these two categories directly into the [NP.NP.NP] pattern. Thus, using syntactic patterns is sufficient for our purposes.

Our experiments revealed additional flexibility in using syntactic patterns. Unlike the OALD codes (which contain at most two arguments or modifiers), the thematic grids consist of up to 4 modifiers. Mapping onto syntactic patterns instead of onto OALD codes allows us to use all arguments in the thematic grids. For example, [NP.NP.PP.from.PP.to] is an example of transitive verb with two prepositional phrases, one beginning with *from* and the other beginning with *to*, as in *"She drove the kids from home to school."*

In the following subsections, we will examine the mapping into these syntactic patterns from: (1) the LCS lexicon; (2) the Minipar codes; and (3) the LDOCE codes.

3.1. Mapping from the LCS Lexicon to Syntactic Patterns

The LCS lexicon consists of verbs grouped into classes based on an adapted version of verb classes (Levin, 1993) along with the thematic grid representations (see (Dorr, 1993; Dorr, 2001)). We automatically assigned syntactic patterns for each verb in the LCS lexicon using its semantic class number and thematic grid. The syntactic patterns we used in our mapping specify prepositions for entries that require them. For example, the grid _ag_th_instr(with) is mapped onto [NP.NP.PP.with] instead of a generic pattern [NP.NP.PP].

More generally, thematic grids contain a list of arguments and modifiers, and they can be obligatory (indicated by an underscore before the role) or optional(indicated by a comma before the role). The arguments can be one of AG, EXP, TH, SRC, GOAL, INFO, PERC, PRED, LOC, POSS, TIME, and PROP. The logical modifiers can be one of MOD-POSS, BEN, INSTR, PURP, MOD-LOC, MANNER, MOD-PRED, MOD-PERC and MOD-PROP. If the argument or the modifier is followed by parenthesis, the corresponding element is a prepositional phrase and its head must be the one specified between the parentheses (if there is nothing between parentheses, PP can begin with any preposition).

Our purpose is to find the set of syntactic patterns for each verb in LCS lexicon using its Levin class and thematic grid. Since each verb can be in many classes and we aim at assigning syntactic patterns based on the semantic classes and thematic grids, there are three possible mapping methodologies:

1. Assign one or more patterns to each class.
2. Assign one or more patterns to each thematic grid.
3. Assign one or more patterns to each pair of class and thematic grid.

The first methodology fails for some classes because the distribution of syntactic patterns over a specific class is not uniform. In other words, attempting to assign only a set of patterns to each class introduces errors because some classes are associated with more than one syntactic frame. For example, class 51.1.d includes three thematic grids: (1) _th,src; (2) _th,src(from); and (3) _th,src(),goal(). We can either assign all patterns for all of these thematic grids to this class or we can choose the most common one. However, both of these approaches introduce errors: The first will generate redundant patterns and the second will assign incorrect patterns to some verbs. (This occurs because, within a class, thematic grids may vary with respect to their optional arguments or the prepositional head associated with arguments or modifiers.)

The second methodology also fails to provide an appropriate mapping. The problem is that some thematic grids correspond to different syntactic patterns in different classes. For example, the thematic grid _th_prop corresponds to 3 different syntactic patterns: (1) [NP.NP] in class 024 and 55.2.a; (2) [NP.ING] in classes 066, 52.b, and 55.2.b; and (3) [NP.INF] in class 005. Although the thematic grid is the same in all of these classes, the syntactic patterns are different.

The final methodology circumvents the two issues presented above (i.e., more than one grid per class and more than one syntactic frame per thematic grid) as follows: If a thematic grid contains an optional argument, we create two mappings for that grid, one in which the optional argument is treated as if it were not there and one in which the argument is obligatory. For example, _ag_th,goal() is mapped onto two patterns [NP.NP] and [NP.NP.PP]. If the number of optional arguments is X, then the maximum number of syntactic patterns for that grid is 2^X (or perhaps smaller than 2^X since some of the patterns may be identical).

Using this methodology, we found the correct mapping for each class and thematic grid pair by examining the verbs in that class and considering all possible syntactic patterns for that pair. This is a many-to-many mapping, i.e. one pattern can be used for different

OALD Code	Syntactic Patterns
I	[NP]
Tn	[NP.NP]
T[n].pr	[NP.NP] and [NP.NP.PP]
Cn.a	[NP.NP.AP]
Cn.n	[NP.NP.NP]
Cn.n/a	[NP.NP.PP.as]
Cn.i	[NP.NP.BARE]
Dn.n	[NP.NP.NP]

Table 3: Mapping From OALD to Syntactic Patterns

LDOCE Code	Syntactic Patterns
I-ABOUT	[NP.PP.about]
I2	[NP.BARE]
L9-WITH	[NP.PP.with]
T1	[NP.NP]
T5	[NP.FIN]
D1	[NP.NP.NP]
D3	[NP.NP.INF]
V4	[NP.NP.ING]

Table 4: Mapping From LDOCE to Syntactic Patterns

pairs and each pair may be associated with more than one pattern. Each verb in each class is assigned the corresponding syntactic patterns according to its thematic grid. Finally, for each verb, we combined all patterns in all classes containing this particular verb in order to generate the lexicon. We will refer to the resulting lexicon as the LCS-based lexicon in Section 4..

3.2. Mapping from Minipar Codes To Syntactic Patterns

Minipar codes are converted straightforwardly into syntactic patterns using the code specification in (Mitten, 1992). An excerpt of the mapping is given in Table 3. This mapping is one-to-many as exemplified by the code $T[n].pr$. Moreover, the set of syntactic patterns extracted from Minipar does not include some patterns such as [NP.PP] (and related patterns) because Minipar does not include modifiers in its code set.

As a result of this mapping, we produced a new lexicon from Minipar entries, where each verb is listed along with the set of syntactic patterns. We will refer to this lexicon as the Minipar-based lexicon in Section 4..

3.3. Mapping from LDOCE Codes to Syntactic Patterns

Similar to the mapping from Minipar to the syntactic patterns, we converted LDOCE codes to syntactic patterns using the code specification in (Procter, 1978). An excerpt of the mapping is given in Table 4.

Each LDOCE code was mapped manually to one or more patterns. LDOCE codes are more refined than the generic OALD codes, but mapping each to syntactic patterns provides an equivalent mediating representation for comparison. For example, LDOCE codes D1-AT and T1-AT are mapped onto [NP.NP.PP.at] by our mapping technique. Again, this is a many-to-many mapping but only a small set of LDOCE codes map to more than one syntactic pattern.

As a result of this mapping, we produced a new lexicon from LDOCE entries, similar to Minipar lexicon. We will refer to this lexicon as the LDOCE-based lexicon in Section 4..

4. Experiments and Results

To measure the effectiveness of our mapping from LCS entries to syntactic patterns, we compared the precision and recall our derived LCS-based syntactic patterns with the precision and recall of Minipar-based syntactic patterns, using LDOCE-based syntactic patterns as our "gold standard".

Each of the three lexicons contains verbs along with their associated syntactic patterns. For experimental purposes, we convert these into pairs. Formally, if a verb v is listed with the patterns $p_1, p_2, \ldots$, we create pairs (v, p_1), (v, p_2) and so on. In addition, we have made the following adjustments to the lexicons, where L is the lexicon under consideration (Minipar or LCS):

1. Given that the number of verbs in each of the two lexicons is different and that neither one completely covers the other, we take only those verbs that occur in both L and LDOCE, for each L, while measuring precision and recall.

2. In the LDOCE- and Minipar-based lexicons, the number of arguments is never greater than 2. Thus, for a fair comparison, we converted the LCS-based lexicon into the same format. For this purpose, we simply omit the arguments after the second one if the pattern contains more than two arguments/modifiers.

3. The prepositions are not specified in Minipar-based lexicon. Thus, we ignore the heads of the prepositions in LCS-based lexicon, i.e., if the pattern includes [PP.$prep$] we take it as a [PP].

Precision and recall are based on the following inputs:

A = Number of pairs in L occurring in LDOCE
B = Number of pairs in L NOT occurring in LDOCE
C = Number of pairs in LDOCE NOT occurring in L

That is, given a syntactic pattern encoded lexicon L, we compute:

(1) The *precision* of $L = \frac{A}{A+B}$;
(2) The *recall* of $L = \frac{A}{A+C}$.

Verbs in LDOCE Lexicon	5648
Verbs in LCS Lexicon	4267
Common verbs in LCS and LDOCE	3757
Pairs in LCS Lexicon	9274
Pairs in LDOCE Lexicon	9200
Pairs in LCS and LDOCE	5654
Verbs fetched completely	1780
Precision	61%
Recall	61%

Table 5: Experiment on LCS-based Lexicon

	All Verbs in Minipar Lexicon	**Common verbs with LCS Lexicon**
Verbs in LDOCE Lexicon	5648	5648
Verbs in Minipar Lexicon	8159	4001
Common verbs in Minipar and LDOCE	5425	3721
Pairs in Minipar Lexicon	10006	7567
Pairs in LDOCE Lexicon	11786	9141
Pairs in Minipar and LDOCE	8014	6124
Verbs fetched completely	3002	1875
Precision	80%	81%
Recall	68%	67%

Table 6: Experiments on Minipar-based Lexicon

We compare two results: one where L is the Minipar-based lexicon and one where L is the LCS-based lexicon. Table 5 gives the number of verbs used in the LCS-based lexicon and the LDOCE-based lexicon, showing the precision and recall. The row showing the number of verbs fetched completely gives the number of verbs in the LCS lexicon which contains all the patterns in the LDOCE entry for the same verb. Both the precision and the recall for LCS-based lexicon with the manually-crafted mapping is 61%.

We did the same experiment for the Minipar-based lexicon in two different ways, first with all the verbs in the Minipar lexicon and then with only the verbs occurring in both the LCS and Minipar lexicons. The second approach is useful for a direct comparison between the Minipar- and LCS-based lexicons. As before, we used the LDOCE-based lexicon as our gold standard. The results are shown in Table 6. The definitions of entries are the same as in Table 5.

The number of Minipar verbs in Minipar occurring in the LCS lexicon is different from the total number of LCS verbs because some LCS verbs (266 of them) do not appear in Minipar lexicon. The results indicate that the Minipar-based lexicon yields much better precision, with an improvement of nearly 25% over the LCS-based lexicon. The recall is low because Minipar does not take modifiers into account most of the time. This results in missing nearly all patterns with PPs, such as [NP.PP] and [NP.NP.PP]. However, the recall achieved is 6% more than the recall for the LCS-based lexicon.

Verbs in LDOCE Lexicon	5648
Verbs in Intersection Lexicon	3623
Common verbs in Int. and LDOCE	3368
Pairs in Intersection Lexicon	4564
Pairs in LDOCE Lexicon	8366
Pairs in Int. and LDOCE	4156
Verbs fetched completely	1265
Precision	91%
Recall	50%

Table 7: Experiment on Intersection Lexicon

Finally, we conducted an experiment to see how the intersection of the Minipar and LCS lexicons compares to the LDOCE-based lexicon. For this experiment, we included only the verbs and patterns occurring in both lexicons. The results are shown in Table 7 in a format similar to previous tables.

The number of common verbs differs from the previous ones because we omit the verbs which do not have any patterns across the two lexicons. The results are not surprising: High precision is achieved because only those patterns that occur in both lexicons are included in the intersection lexicon; thus, the total number of pairs is reduced significantly. For the same reason, the recall is significantly reduced.

The highest precision is achieved by the intersection of two lexicons, but at the expense of recall. We found that the precision was higher for Minipar than for the LCS lexicon, but when we examined this in more detail, we found that this was almost entirely due to "double counting" of entries with optional modifiers in the LCS-based lexicon. For example, the single LCS-based grid _ag_th,instr(with) corresponds to two syntactic patterns, [NP.NP] and [NP.NP.PP], while LDOCE views these as the single pattern [NP.NP]. Specifically, 53% of the non-matching LCS-based patterns are [NP.NP.PP]—and 93% of these co-occur with [NP.NP]. Similarly, 13% of the non-matching LCS-based patterns are pattern [NP.PP]—and 80% of these co-occur with [NP].

This is a significant finding, as it reveals that our precision is spuriously low in our comparison with the "gold standard." In effect, we should be counting the LCS-based pattern [NP.NP.PP]/[NP.NP] to be a match against the LDOCE-based pattern [NP.NP]—which is a fairer comparison since neither LDOCE nor Minipar takes modifiers into account. (We henceforth refer to LCS-based the co-occurring patterns

	Minipar Lexicon (All verbs in Minipar Lexicon)	Minipar Lexicon (Common verbs with LCS Lexicon)	LCS Lexicon	Intersection of Minipar and LCS Lexicons
Precision	80%	81%	61%	91%
Enhanced Precision	81%	82%	80%	91%
Recall	68%	67%	61%	50%

Table 8: Precision and Recall Summary: Minipar- and LCS-based Lexicons

[NP.NP.PP]/[NP.NP] and [NP.PP]/[NP] as overlapping pairs.) To observe the degree of the impact of optional modifiers, we computed another precision value for the LCS-based lexicon by counting overlapping patterns once instead of twice. With this methodology, we achieved 80% (enhanced) precision. This precision value is nearly same as the value achieved with the current Minipar lexicon. Table 8 summarizes all results in terms of precision and recall.

The enhanced precision is an important and accurate indicator of the effectiveness of our approach, given that overlapping patterns arise because of (optional) modifiers. When we ignore those modifiers during our mapping process, we achieve nearly the same precision and recall with the current Minipar lexicon, which also ignores the modifiers in its code set. Moreover, overlapping patterns in our LCS-based lexicon do not affect the performance of the parser, other than to induce a more sophisticated handling of modifiers (which presumably would increase the precision numbers, if we had access to a "gold standard" that includes modifiers). For example, Minipar attaches modifiers at the clausal level instead of at the verbal level even in cases where the modifier is obviously verbal—as it would be in the LCS-based version of the parse in the sentence *She rolled the dough [PP into cookie shapes].*

5. Ongoing Work: Automatic Generation of Syntactic Patterns

The lexicon derived from the hand-crafted mapping between the LCS lexicon and the syntactic patterns is comparable to the current Minipar lexicon. However, the mapping required a great deal of human effort, since each semantic verb class must be examined by hand in order to identify appropriate syntactic patterns. The process is error-prone, laborious, and time-intensive (approximately 3-4 person-months). Moreover, it requires that the mapping be done again by a human every time the LCS lexicon is updated.

In a recent experiment, we developed an automated mapping (in 2 person-weeks) that takes into account both semantic roles and some additional features stored in the LCS database, without reference to the class number. The mapping is based primarily on the thematic role, however in some situations the thematic roles themselves are not sufficient to determine the type of the argument. In such cases, the correct form is assigned using featural information associated with that specific verb in the LCS database.

Table 10 summarizes the automated mapping rules. The thematic role "prop" is an example of a case where featural information is necessary (e.g., (cform inf)), as there are five different patterns to choose from for this thematic role. Similarly, whether a "pred" role is an NP or AP is determined by featural information. For example, this role becomes an AP for the verb *behave* in class 29.6.a while it is mapped onto an NP for the verb *carry* in class 54.2. In the cases where the syntactic pattern is ambiguous and there is no specification for the verbs, default values are used for the mapping: BARE for "prop", AP for "pred" and NP for "perc".

Syntactic patterns for each thematic grid are computed by combining the results of the mapping from each thematic role in the grid to a syntactic pattern, one after another. If the grid includes optional roles, every possibility is explored and the syntactic patterns for each of them is included in the whole list of patterns for that grid. For example, the syntactic patterns for _ag_th,instr(with) include the patterns for both _ag_th and _ag_th_instr(with), which are [NP.NP] and [NP.NP.PP.with].

Note that this approach eliminates the need for using the same syntactic patterns for all verbs in a specific class: Verbs in the same class can be assigned different syntactic patterns with the help of additional features in the database. Thus, we need not rely on the semantic class number at all during this mapping. We can easily update the resulting lexicons when there is any change on the semantic classes or thematic grids of some verbs.

This experiment resulted in a parsing lexicon that has virtually the same precision/recall as that of the manually generated LCS-based lexicon above. (See Table 9.) As in the case of the manually generated mappings, the enhanced precision is 80%, which is

Verbs in LDOCE Lexicon	5648
Verbs in LCS Lexicon	4267
Common verbs in LCS and LDOCE	3757
Pairs in LCS Lexicon	9253
Pairs in LDOCE Lexicon	9200
Pairs in LCS and LDOCE	5634
Verbs fetched completely	1781
Precision	61%
Enhanced Precision	80%
Recall	61%

Table 9: Precision and Recall of Automatic Generation of Syntactic Patterns

Thematic Role	**Syntactic Patterns**
particle	PREP
prop(...), mod-prop(...), info(...)	FIN or INF or ING or PP
all other role(...)	PP
th, exp, info	FIN or INF or ING or NP
prop	NP or ING or INF or FIN or BARE
pred	AP or NP
perc	[NP.ING] or [NP.BARE]
all other roles	NP

Table 10: Syntactic Patterns Corresponding to Thematic Roles

only 1-2% lower than that of the current Minipar-based lexicon.

Our approach demonstrates that examination of thematic-role and featural information in the LCS-based lexicon is sufficient for executing this mapping automatically. Automating our approach gives us the flexibility of re-running the program if the structure of the database changes (e.g., an LCS representation is modified or class membership changes) and of porting to a new language with minimal effort.

6. Discussion

In all experiments reported above, both the LCS- and Minipar-based lexicons yield low recall values. Upon further investigation, we found that LDOCE is too specific in assigning codes to verbs. Most of the patterns associated with the verbs are rare—cases not considered in the LCS- and Minipar-based lexicons. Because of that, we believe that the recall values will improve if we take only a subset of LDOCE-based lexicon, e.g., those associated with the most frequent verb-pattern pairs in a large corpus. This is a future research direction considered in the next section.

The knowledgeable reader may question the mapping of a Levin-style lexicon into syntactic codes, given that Levin's original proposal is to investigate verb meaning through examination of syntactic patterns, or *alternations*, in the first place. As alluded to in Section 1., there are several ways in which this database has become more than just a "semantified" version of a syntactic framework; we elaborate on this further here.

Levin's original framework omitted a large number of verbs—and verb senses for existing Levin verbs—which we added to the database by semi-automatic techniques. Her original framework contained 3024 verbs in 192 classes numbering between 9.1 and 57—a total of 4186 verb entries. These were grouped together primarily by means of syntactic alternations. Our augmented database contains 4432 verbs in 492 classes with more specific numbering (e.g., "51.3.2.a.ii") including additional class numbers for new classes that Levin did not include in her work (between 000 and 026)—a total of 9844 verb entries. These were categorized according to semantic information (using WordNet synsets coupled with syntactic filtering) (Dorr, 1997)—not syntactic alternations.

An example of an entry that we added to the database is the verb *oblige*. We have assigned a semantic representation and thematic grid to this verb, creating a new class 002—which we call *Coerce Verbs*—corresponding to verbs whose underlying meaning corresponds to "force to act". Because Levin's repository omits verbs taking clausal complements, several other verbs with a similar meaning fell into this class (e.g., *coerce*, *compel*, *persuade*) including some that were already included in the original system, but not in this class (e.g., *ask*). Thus, the LCS Database contains 50% more verbs and twice as many verb entries since the original framework of Levin. The result is that we can now parse constructions such as *She compelled him to eat* and *She asked him to eat*, which would not have been analyzable had we compiled our parsing lexicon on the basis of Levin's classes alone.

Levin's original proposal also does not contain semantic representations or thematic grids. When we built the LCS database, we examined each verb class carefully by hand to determine the underlying components of meaning unifying the members of that class. For example, the LCS representation that we generated for verbs in the *put* class includes components of meaning corresponding to "spatial placement in some manner," thus covering *dangle*, *hang*, *suspend*, etc.

From these hand-generated LCS representations, we derived our thematic grids—the same ones that are mapped onto our syntactic patterns. For example, position 1 (the highest leftmost argument in the LCS) is always mapped into the agent role of the thematic grid. The grids are organized into a thematic hierarchy that provides the basis for determining argument assignments, thus enhancing the generation process in ways that could not have been done previously with Levin's classes alone—e.g., producing constructions like *John sent a book to Paul* instead of constructions like *The book sent John to Paul*. Although the value of the thematic hierarchy seems most relevant to generation, the overall semantic/thematic hierarchical organization enables the automatic construction of lexicons that are equally suitable for both parsing and generation, thus reducing our overall lexical acquisition effort for both processes.

Beyond the above considerations, the granularity of the original Levin framework also was not adequate for our interlingual MT and lexical acquisition efforts. Our augmented form of this repository has brought about a more refined classification in which we are able to accommodate aspectual distinctions. We encode knowledge about aspectual features (e.g., *telicity*) in our LCS representations, thus sub-dividing the classes into more specific sub-classes. The tests used for this sub-division are purely *semantic* in nature, not syntactic. An example is the Dowty-style test "*He was X-ing* entails *He has X-ed*" (Dowty, 1979), where X is atelic (as in *run*) only if this entailment is considered valid by a human—and telic otherwise (as in *win*).

The inclusion of this type of knowledge allows us to refine Levin's classification significantly. An example is Class 35.6—*Ferret Verbs*: In Levin's original framework, this class conflated verbs occurring in different aspectual categories. Using the semantic tests above, we found that, in fact, these verbs should be divided as follows (Olsen et al., 1997):

Ferret Verbs: nose ferret tease (telic); seek (atelic)

The implication of this division for parsing is that the verbal arguments are constrained in a way that was not available to us in the original Levin-style classification—thus easing the job of the parser in choosing attachment points:

Telic:

$*$He ferreted the truth from him.

He ferreted the truth out of him

Atelic:

He sought the truth from him.

$*$He sought the truth out of him

Finally, Levin makes no claims as to the applicability of the English classes to other languages. Orienting our LCS database more toward semantic (aspectual) features rather than syntactic alternations has brought us closer to an interlingual representation that has now been demonstrably ported (quickly) to multiple languages including Arabic, Chinese, and Spanish. For example, telicity has been shown to be a crucial deciding feature in translating between divergence languages (Olsen et al., 1998), as in the translation of English *run across* as Spanish *cruzar corriendo*.

To summarize, our work is intended to: (1) Investigate the realization of a parsing lexicon from an LCS database that has developed from extensive semantic enhancements to an existing framework of verb classes and (2) Automate this technique so that it is directly applicable to LCS databases in other languages.

7. Future Work and Conclusions

Our ongoing work involves the following:

1. Using a subset of LDOCE-based lexicon by taking only the most frequent verb-pattern pairs in a big corpus: We expect that this approach will produce more realistic recall values.

2. Creating parsing lexicons for different languages: Once we have an automated mapping from the semantic lexicon to the set of syntactic patterns, we can use this method to create parsing lexicons from semantic lexicons that we already have available in other languages (Chinese, Spanish and Arabic).

3. Integration of these parsing lexicons in ongoing machine translation work (Habash and Dorr, 2001): We will feed the created lexicons into a parser and examine how successful the lexicons are. The same lexicons will also be used in our current clustering project.

Some of the ideas mentioned above are explored in detail in (Ayan and Dorr, 2002).

We conclude that it is possible to produce a parsing lexicon by projecting from LCS-based lexical entries—achieving precision and recall on a par with a syntactic lexicon (Minipar) encoded by hand specifically for English. The consequence of this result is that, as semantic lexicons become increasingly available for multiple languages (ours are now available in English, Chinese, and Arabic), we are able to produce parsing lexicons automatically for each language.

Acknowledgments

This work has been supported, in part, by ONR MURI Contract FCPO.810548265 and Mitre Contract 010418-7712.

8. References

Necip Fazil Ayan and Bonnie J. Dorr. 2002. Creating Parsing Lexicons From Semantic Lexicons Automatically and Its Applications. Technical report, University of Maryland, College Park, MD. Technical Report: LAMP-TR-084, CS-TR-4352, UMIACS-TR-2002-32.

Michael Brent. 1993. From Grammar to Lexicon: Unsupervised Learning of Lexical Syntax. *Computational Linguistics*, 19(2):243–262.

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, DC.

J. Carroll and C. Grover. 1989. The Derivation of a Large Computational Lexicon for English from LDOCE. In B. Boguraev and Ted Briscoe, editors, *Computational lexicography for natural language processing*, pages 117–134. Longman, London.

Bonnie J. Dorr. 1993. *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, MA.

Bonnie J. Dorr. 1997. Large-Scale Dictionary Construction for Foreign Language Tutoring and Interlingual Machine Translation. *Machine Translation*, 12(4):271–322.

Bonnie J. Dorr. 2001. LCS Verb Database. Technical Report Online Software Database, University of Maryland, College Park, MD. http://www.umiacs.umd.edu/˜bonnie/-LCS_Database_Documentation.html.

David Dowty. 1979. *Word Meaning in Montague Grammar*. Reidel, Dordrecht.

Dania Egedi and Patrick Martin. 1994. A Freely Available Syntactic Lexicon for English. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, Nara, Japan.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. Comlex Syntax: Building a Computational Lexicon. In *Proceedings of the COLING*, Kyoto.

Nizar Habash and Bonnie Dorr. 2001. Large-Scale Language Independent Generation Using Thematic Hierarchies. In *Proceedings of MT Summit VIII, Santiago de Compostella, Spain*.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL.

Dekang Lin. 1993. Principle-Based Parsing without Overgeneration. In *Proceedings of ACL-93*, pages 112–120, Columbus, Ohio.

Dekang Lin. 1998. Dependency-Based Evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*, Granada, Spain, May.

Christopher D. Manning. 1993. Automatic Acquisition of a Large Subcategorization Dictionary from Corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242, Columbus, Ohio.

R. Mitten. 1992. *Computer-Usable Version of Oxford Advanced Learner's Dictionary of Current English*. Oxford Text Archive.

Mari Broman Olsen, Bonnie J. Dorr, and Scott C. Thomas. 1997. Toward Compact Monotonically Compositional Interlingua Using Lexical Aspect. In *Proceedings of the Workshop on Interlinguas in MT, MT Summit, New Mexico State University Technical Report MCCS-97-314*, pages 33–44, San Diego, CA, October. Also available as UMIACS-TR-97-86, LAMP-TR-012, CS-TR-3858, University of Maryland.

Mari Broman Olsen, Bonnie J. Dorr, and Scott C. Thomas. 1998. Enhancing Automatic Acquisition of Thematic Structure in a Large-Scale Lexicon for Mandarin Chinese. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas, AMTA-98, in Lecture Notes in Artificial Intelligence, 1529*, pages 41–50, Langhorne, PA, October 28–31.

P. Procter. 1978. *Longman Dictionary of Contemporary English*. Longman, London.

Suzanne Stevenson and Paola Merlo. 2002a. A Multilingual Paradigm for Automatic Verb Classification. In *Proceedings of Association of Computational Linguistics*, Philadelphia, PA.

Suzanne Stevenson and Paola Merlo. 2002b. Automatic verb classification using distributions of grammatical features. In *Proceedings of the 9th Conference of the European Chapter of ACL*, pages 45–52, Bergen, Norway.

Building Thematic Lexical Resources by Bootstrapping and Machine Learning

Alberto Lavelli*, Bernardo Magnini*, Fabrizio Sebastiani†

*ITC-irst
Via Sommarive, 18 – Località Povo
38050 Trento, Italy
{lavelli,magnini}@itc.it

†Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
fabrizio@iei.pi.cnr.it

Abstract

We discuss work in progress in the semi-automatic generation of *thematic lexicons* by means of *term categorization*, a novel task employing techniques from information retrieval (IR) and machine learning (ML). Specifically, we view the generation of such lexicons as an iterative process of learning previously unknown associations between terms and *themes* (i.e. disciplines, or fields of activity). The process is iterative, in that it generates, for each c_i in a set $C = \{c_1, \ldots, c_m\}$ of themes, a sequence $L_0^i \subseteq L_1^i \subseteq \ldots \subseteq L_n^i$ of lexicons, bootstrapping from an initial lexicon L_0^i and a set of text corpora $\Theta = \{\theta_0, \ldots, \theta_{n-1}\}$ given as input. The method is inspired by *text categorization*, the discipline concerned with labelling natural language texts with labels from a predefined set of themes, or categories. However, while text categorization deals with documents represented as vectors in a space of terms, we formulate the task of term categorization as one in which terms are (dually) represented as vectors in a space of documents, and in which terms (instead of documents) are labelled with themes. As a learning device, we adopt *boosting*, since (a) it has demonstrated state-of-the-art effectiveness in a variety of text categorization applications, and (b) it naturally allows for a form of "data cleaning", thereby making the process of generating a thematic lexicon an iteration of generate-and-test steps.

1. Introduction

The generation of *thematic lexicons* (i.e. lexicons consisting of specialized terms, all pertaining to a given theme or discipline) is a task of increased applicative interest, since such lexicons are of the utmost importance in a variety of tasks pertaining to natural language processing and information access.

One of these tasks is to support text search and other information retrieval applications in the context of thematic, "vertical" portals (aka *vortals*)[1]. Vortals are a recent phenomenon in the World Wide Web, and have grown out of the users' needs for directories, services and information resources that are both rich in information and specific to their interests. This has led to Web sites that specialize in aggregating market-specific, "vertical" content and information. Actually, the evolution from the generic portals of the previous generation (such as Yahoo!) to today's vertical portals is just natural, and is no different from the evolution that the publishing industry has witnessed decades ago with the creation of specialized magazines, targeting specific categories of readers with specific needs. To read about the newest developments in ski construction technology, skiers read specialty magazines about skiing, and not generic newspapers, and skiing magazines is also where advertisers striving to target skiers place their ads in order to be the most effective. Vertical portals are the future of commerce and information seeking on the Internet, and supporting sophisticated information access capabilities by means of thematic lexical resources is thus of the utmost importance.

Unfortunately, the generation of thematic lexicons is expensive, since it requires the intervention of specialized manpower, i.e. lexicographers and domain experts working together. Besides being expensive, such a manual approach does not allow for fast response to rapidly emerging needs. In an era of frantic technical progress new disciplines emerge quickly, while others disappear as quickly; and in an era of evolving consumer needs, the same goes for new market niches. There is thus a need of cheaper and faster methods for answering application needs than manual lexicon generation. Also, as noted in (Riloff and Shepherd, 1999), the manual approach is prone to errors of omission, in that a lexicographer may easily overlook infrequent, non-obvious terms that are nonetheless important for many tasks.

Many applications also require that the lexicons be not only thematic, but also tailored to the specific data tackled in the application. For instance, in query expansion (automatic (Peat and Willett, 1991) or interactive (Sebastiani, 1999)) for information retrieval systems addressing thematic document collections, terms synonymous or quasi-synonymous to the query terms are added to the query in order to retrieve more documents. In this case, the added terms should occur in the document collection, otherwise they are useless, and the relevant terms which occur in the document collection should potentially be added. That is, for this application the ideal thematic lexicon should contain all and only the technical terms present in the document

[1]See e.g. `http://www.verticalportals.com/`

collection under consideration, and should thus be generated directly from this latter.

1.1. Our proposal

In this paper we propose a methodology for the semi-automatic generation of thematic lexicons from a corpus of texts. This methodology relies on *term categorization*, a novel task that employs a combination of techniques from information retrieval (IR) and machine learning (ML). Specifically, we view the generation of such lexicons as an iterative process of learning previously unknown associations between terms and *themes* (i.e. disciplines, or fields of activity)[2]. The process is iterative, in that it generates, for each c_i in a set $C = \{c_1, \ldots, c_m\}$ of predefined themes, a sequence $L^i_0 \subseteq L^i_1 \subseteq \ldots \subseteq L^i_n$ of lexicons, bootstrapping from a lexicon L^i_0 given as input. Associations between terms and themes are learnt from a sequence $\Theta = \{\theta_0, \ldots, \theta_{n-1}\}$ of sets of documents (hereafter called *corpora*); this allows to enlarge the lexicon as new corpora from which to learn become available. At iteration y, the process builds the lexicons $L_{y+1} = \{L^1_{y+1}, \ldots, L^m_{y+1}\}$ for all the themes $C = \{c_1, \ldots, c_m\}$ in parallel, from the same corpus θ_y. The only requirement on θ_y is that at least some of the terms in each of the lexicons in $L_y = \{L^1_y, \ldots, L^m_y\}$ should occur in it (if none among the terms in a lexicon L^j_y occurs in θ_y, then no new term is added to L^j_y in iteration y).

The method we propose is inspired by *text categorization*, the activity of automatically building, by means of machine learning techniques, *automatic text classifiers*, i.e. programs capable of labelling natural language texts with (zero, one, or several) thematic categories from a predefined set $C = \{c_1, \ldots, c_m\}$ (Sebastiani, 2002). The construction of an automatic text classifier requires the availability of a corpus $\psi = \{\langle d_1, C_1\rangle, \ldots, \langle d_h, C_h\rangle\}$ of preclassified documents, where a pair $\langle d_j, C_j\rangle$ indicates that document d_j belongs to all and only the categories in $C_j \subseteq C$. A general inductive process (called the *learner*) automatically builds a classifier for the set C by learning the characteristics of C from a *training set* $Tr = \{\langle d_1, C_1\rangle, \ldots, \langle d_g, C_g\rangle\} \subset \psi$ of documents. Once a classifier has been built, its effectiveness (i.e. its capability to take the right categorization decisions) may be tested by applying it to the *test set* $Te = \{\langle d_{g+1}, C_{g+1}\rangle, \ldots, \langle d_h, C_h\rangle\} = \psi - Tr$ and checking the degree of correspondence between the decisions of the automatic classifier and those encoded in the corpus.

While the purpose of text categorization is that of classifying documents represented as vectors in a space of terms, the purpose of term categorization, as we formulate it, is (dually) that of classifying terms represented as vectors in a space of documents. In this task terms are thus items that may belong, and must thus be assigned, to (zero, one, or several) themes belonging to a predefined set. In other words, starting from a set Γ^i_y of preclassified terms, a new set of terms Γ^i_{y+1} is classified, and the terms in Γ^i_{y+1} which are deemed to belong to c_i are added to L^i_y to yield L^i_{y+1}. The set Γ^i_y is composed of lexicon L^i_y, acting as the set of "positive examples", plus a set of terms known not to belong to c_i, acting as the set of "negative examples".

For input to the learning device and to the term classifiers that this will eventually build, we use "bag of documents" representations for terms (Salton and McGill, 1983, pages 78–81), dual to the "bag of terms" representations commonly used in text categorization.

As the learning device we adopt ADABOOST.MH^{KR} (Sebastiani et al., 2000), a more efficient variant of the ADABOOST.MH^{R} algorithm proposed in (Schapire and Singer, 2000). Both algorithms are an implementation of *boosting*, a method for supervised learning which has successfully been applied to many different domains and which has proven one of the best performers in text categorization applications so far. Boosting is based on the idea of relying on the collective judgment of a committee of classifiers that are trained sequentially; in training the k-th classifier special emphasis is placed on the correct categorization of the training examples which have proven harder for (i.e. have been misclassified more frequently by) the previously trained classifiers.

We have chosen a boosting approach not only because of its state-of-the-art effectiveness, but also because it naturally allows for a form of "data cleaning", which is useful in case a lexicographer wants to check the results and edit the newly generated lexicon. That is, in our term categorization context it allows the lexicographer to easily inspect the classified terms for possible misclassifications, since at each iteration y the algorithm, apart from generating the new lexicon L^i_{y+1}, ranks the terms in L^i_y in terms of their "hardness", i.e. how successful have been the generated classifiers at correctly recognizing their label. Since the highest ranked terms are the ones with the highest probability of having been misclassified in the previous iteration (Abney et al., 1999), the lexicographer can examine this list starting from the top and stopping where desired, removing the misclassified examples. The process of generating a thematic lexicon then becomes an iteration of generate-and-test steps.

This paper is organized as follows. In Section 2. we describe how we represent terms by means of a "bag of documents" representation.. For reasons of space we do not describe ADABOOST.MH^{KR}, the boosting algorithm we employ for term classification; see the extended paper for details (Lavelli et al., 2002). Section 3.1. discusses how to combine the indexing tools introduced in Section 2. with the boosting algorithm, and describes the role of the lexicographer in the iterative generate-and-test cycle. Section 3.2. describes the results of our preliminary experiments. In Section 4. we review related work on the automated generation of lexical resources, and spell out the differences between our and existing approaches. Section 5. concludes, pointing to avenues for improvement.

[2]We want to point out that our use of the word "term" is somehow different from the one often used in natural language processing and terminology extraction (Kageura and Umino, 1996), where it often denotes a *sequence* of lexical units expressing a concept of the domain of interest. Here we use this word in a neutral sense, i.e. without making any commitment as to its consisting of a single word or a sequence of words.

2. Representing terms in a space of documents

2.1. Text indexing

In text categorization applications, the process of building internal representations of texts is called *text indexing*. In text indexing, a document d_j is usually represented as a vector of term *weights* $\vec{d_j} = \langle w_{1j}, \ldots, w_{rj} \rangle$, where r is the cardinality of the *dictionary* and $0 \leq w_{kj} \leq 1$ represents, loosely speaking, the contribution of t_k to the specification of the semantics of d_j. Usually, the dictionary is equated with the set of *terms* that occur at least once in at least α documents of Tr (with α a predefined threshold, typically ranging between 1 and 5).

Different approaches to text indexing may result from different choices (i) as to what a term is and (ii) as to how term weights should be computed. A frequent choice for (i) is to use single words (minus stop words, which are usually removed prior to indexing) or their stems, although some researchers additionally consider noun phrases (Lewis, 1992) or "bigrams" (Caropreso et al., 2001). Different "weighting" functions may be used for tackling issue (ii), either of a probabilistic or of a statistical nature; a frequent choice is the *normalized* $tfidf$ function (see e.g. (Salton and Buckley, 1988)), which provides the inspiration for our "term indexing" methodology spelled out in Section 2.2..

2.2. Abstract indexing and term indexing

Text indexing may be viewed as a particular instance of *abstract indexing*, a task in which abstract objects are represented by means of abstract features, and whose underlying metaphor is, by and large, that the semantics of an object corresponds to the *bag of features* that "occur" in it[3]. In order to illustrate abstract indexing, let us define a *token* τ to be a specific occurrence of a given feature $f(\tau)$ in a given object $o(\tau)$, let T be the set of all tokens occurring in any of a set of objects O, and let F be the set of features of which the tokens in T are instances. Let us define the *feature frequency* $ff(f_k, o_j)$ of a feature f_k in an object o_j as

$$ff(f_k, o_j) = |\{\tau \in T \mid f(\tau) = f_k \ \wedge \ o(\tau) = o_j\}| \quad (1)$$

We next define the *inverted object frequency* $iof(f_k)$ of a feature f_k as

$$\begin{aligned} iof(f_k) &= \qquad (2) \\ &= \log \frac{|O|}{|\{o_j \in O \mid \exists \tau \in T : f(\tau) = f_k \ \wedge \ o(\tau) = o_j\}|} \end{aligned}$$

and the *weight* $w(f_k, o_j)$ of feature f_k in object o_j as

$$\begin{aligned} w_{kj} &= w(f_k, o_j) = \qquad (3) \\ &= \frac{ff(f_k, o_j) \cdot iof(f_k)}{\sqrt{\sum_{s=1}^{|F|} (ff(f_s, o_j) \cdot iof(f_s))^2}} \end{aligned}$$

[3]"Bag" is used here in its set-theoretic meaning, as a synonym of *multiset*, i.e. a set in which the same element may occur several times. In text indexing, adopting a "bag of words" model means assuming that the number of times that a given word occurs in the same document is semantically significant. "Set of words" models, in which this number is assumed not significant, are thus particular instances of bag of words models.

We may consider the $w(f_k, o_j)$ function of Equation (3) as an *abstract indexing function*; that is, different instances of this function are obtained by specifying different choices for the set of objects O and set of features F.

The well-known text indexing function $tfidf$, mentioned in Section 2.1., is obtained by equating O with the training set of documents and F with the dictionary; T, the set of occurrences of elements of F in the elements of O, thus becomes the set of term occurrences.

Dually, a term indexing function may be obtained by switching the roles of F and O, i.e. equating F with the training set of documents and O with the dictionary; T, the set of occurrences of elements of F in the elements of O, is thus again the set of term occurrences (Schäuble and Knaus, 1992; Sheridan et al., 1997).

It is interesting to discuss the kind of intuitions that Equations (1), (2) and (3) embody in the dual cases of text indexing and term indexing:

- Equation (1) suggests that when a feature occurs multiple times in an object, the feature characterizes the object to a higher degree. In text indexing, this indicates that the more often a term occurs in a document, the more it is representative of its content. In term indexing, this indicates that the more often a term occurs in a document, the more the document is representative of the content of the term.

- Equation (2) suggests that the fewer the objects a feature occurs in, the more representative it is of the content of the objects in which it occurs. In text indexing, this means that terms that occur in too many documents are not very useful for identifying the content of documents. In term indexing, this means that the more terms a document contains (i.e. the longer it is), the less useful it is for characterizing the semantics of a term it contains.

- The intuition ("length normalization") that supports Equation (3) is that weights computed by means of $ff(f_k, o_j) \cdot iof(f_k)$ need to be normalized in order to prevent "longer objects" (i.e. ones in which many features occur) to emerge (e.g. to be scored higher in document-document similarity computations) just because of their length and not because of their content. In text indexing, this means that longer documents need to be deemphasized. In term indexing, this means instead that terms that occur in many documents need to be deemphasized[4].

It is also interesting to note that any program or data structure that implements $tfidf$ for text indexing may be used straightaway, with no modification, for term indexing: one needs only to feed the program with the terms in place of the documents and viceversa.

[4]Incidentally, it is interesting to note that in switching from text indexing to term indexing, Equations (2) and (3) switch their roles: the intuition that terms occurring in many documents should be deemphasized is implemented in Equation (2) in text indexing and Equation (3) in term indexing, while the intuition that longer documents need to be deemphasized is implemented in Equation (3) in text indexing and Equation (2) in term indexing.

3. Generating thematic lexicons by bootstrapping and learning

3.1. Operational methodology

We are now ready to describe the overall process that we will follow for the generation of thematic lexicons. The process is iterative: we here describe the y-th iteration. We start from a set of thematic lexicons $L_y = \{L_y^1, \ldots, L_y^m\}$, one for each theme in $C = \{c_1, \ldots, c_m\}$, and from a corpus θ_y. We index the terms that occur in θ_y by means of the term indexing technique described in Section 2.2.; this yields, for each term t_k, a representation consisting of a vector of weighted documents, the length of the vector being $r = |\theta_y|$.

By using $L_y = \{L_y^1, \ldots, L_y^m\}$ as a training set, we then generate m classifiers $\Phi_y = \{\Phi_y^1, \ldots, \Phi_y^m\}$ by applying the AdaBoost.MH^{KR} algorithm. While generating the classifiers, AdaBoost.MH^{KR} also produces, for each theme c_i, a ranking of the terms in L_y^i in terms of how hard it was for the generated classifiers to classify them correctly, which basically corresponds to their probability of being misclassified examples. The lexicographer can then, if desired, inspect L_y and remove the misclassified examples, if any (possibly rerunning, especially if these latter were a substantial number, AdaBoost.MH^{KR} on the "cleaned" version of L_y). At this point, the terms occurring in θ_y that AdaBoost.MH^{KR} has classified under c_i are added (possibly, after being checked by the lexicographer) to L_y^i, yielding L_{y+1}^i. Iteration $y + 1$ can then take place, and the process is repeated again.

Note that an alternative approach is to involve the lexicographer only after the last iteration, and not after each iteration. For instance, Riloff and Shepherd (Riloff and Shepherd, 1999) perform several iterations, at each of which they add to the training set (without human intervention) the new items that have been attributed to the category with the highest confidence. After the last iteration, a lexicographer inspects the list of added terms and decides which one to remove, if any. This latter approach has the advantage of requiring the intervention of the lexicographer only once, but has the disadvantage that spurious terms added to lexicon at early iterations can cause, if not promptly removed, new spurious ones to be added in the next iterations, thereby generating a domino effect.

3.2. Experimental methodology

The process we have described in Section 3.1. is the one that we would apply in an operational setting. In an experimental setting, instead, we are also interested in evaluating the effectiveness of our approach on a benchmark. The difference with the process outlined in Section 3.1. is that at the beginning of the process the lexicon L_y is split into a training set and a test set; the classifiers are learnt from the training set, and are then tested on the test set by checking how good they are at extracting the terms in the test set from the corpus θ_y. Of course, in order to guarantee a fair evaluation, the terms that never occur in θ_y are removed from the test set, since there is no way that the algorithm (or any other algorithm that extracts terms from a corpus) could possibly guess them.

Category c_i		expert judgments YES	expert judgments NO
classifier judgments	**YES**	TP_i	FP_i
	NO	FN_i	TN_i

Table 1: The contingency table for category c_i. Here, FP_i (*false positives wrt* c_i) is the number of test terms incorrectly classified under c_i; TN_i (*true negatives wrt* c_i), TP_i (*true positives wrt* c_i) and FN_i (*false negatives wrt* c_i) are defined accordingly.

We will comply with standard text categorization practice in evaluating term categorization effectiveness by a combination of *precision* (π), the percentage of positive categorization decisions that turn out to be correct, and *recall* (ρ), the percentage of positive, correct categorization decisions that are actually taken. Since most classifiers can be tuned to emphasize one at the expense of the other, only combinations of the two are usually considered significant. Following common practice, as a measure combining the two we will adopt their harmonic mean, i.e. $F_1 = \frac{2\pi\rho}{\pi+\rho}$. Effectiveness will be computed with reference to the contingency table illustrated in Table 1. When effectiveness is computed for several categories, the results for individual categories must be averaged in some way; we will do this both by *microaveraging* ("categories count proportionally to the number of their positive training examples"), i.e.

$$\pi^\mu = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{|\mathcal{C}|}(TP_i + FP_i)}$$

$$\rho^\mu = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FN_i)}$$

and by *macroaveraging* ("all categories count the same"), i.e.

$$\pi^M = \frac{\sum_{i=1}^{|\mathcal{C}|} \pi_i}{m} \qquad \rho^M = \frac{\sum_{i=1}^{m} \rho_i}{m}$$

Here, "μ" and "M" indicate microaveraging and macroaveraging, respectively, while the other symbols are as defined in Table 1. Microaveraging rewards classifiers that behave well on *frequent categories* (i.e. categories with many positive test examples), while classifiers that perform well also on infrequent categories are emphasized by macroaveraging. Whether one or the other should be adopted obviously depends on the application.

3.3. Our experimental setting

We now describe the resources we have used in our experiments.

3.3.1. The corpora

As the corpora $\Theta = \{\theta_1, \ldots, \theta_n\}$, we have used various subsets of the Reuters Corpus Volume I (RCVI), a corpus of documents recently made available by Reuters[5] for text categorization experimentation and consisting of about 810,000 news stories. Note that, although the texts of RCVI

[5]http://www.reuters.com/

are labelled by thematic categories, we have not made use of such labels (not it would have made much sense to use them, given that these categories are different from the ones we are working with); the reasons we have chosen this corpus instead of other corpora of unlabelled texts are inessential.

3.3.2. The lexicons

As the thematic lexicons we have used subsets of an extension of WordNet, that we now describe.

WordNet (Fellbaum, 1998) is a large, widely available, non-thematic, monolingual, machine-readable dictionary in which sets of synonymous words are grouped into synonym sets (or *synsets*) organized into a directed acyclic graph. In this work, we will always refer to WordNet version 1.6.

In WordNet only a few synsets are labelled with thematic categories, mainly contained in the glosses. This limitation is overcome in WordNetDomains, an extension of WordNet described in (Magnini and Cavaglià, 2000) in which each synset has been labelled with one or more from a set of 164 thematic categories, called *domains*[6]. The 164 domains of WordNetDomains are a subset of the categories belonging to the classification scheme of Dewey Decimal Classification (DDC (Mai Chan et al., 1996)); example domains are ZOOLOGY, SPORT, and BASKETBALL.

These 164 domains have been chosen from the much larger set of DDC categories since they are the most popular labels used in dictionaries for sense discrimination purposes. Domains have long been used in lexicography (where they are sometimes called *subject field codes* (Procter, 1978)) to mark technical usages of words. Although they convey useful information for sense discrimination, they typically tag only a small portion of a dictionary. WordNetDomains extends instead the coverage of domain labels to an entire, existing lexical database, i.e. WordNet.

A domain may include synsets of different syntactic categories: for instance, the MEDICINE domain groups together senses from Nouns, such as `doctor#1` (the first among several senses of the word "doctor") and `hospital#1`, and from Verbs, such as `operate#7`. A domain may include senses from different WordNet subhierarchies. For example, SPORT contains senses such as `athlete#1`, which descends from `life_form#1`; `game_equipment#1`, from `physical_object#1`; `sport#1`, from `act#2`; and `playing_field#1`, from `location#1`. Note that domains may group senses of the same word into thematic clusters, with the side effect of reducing word polysemy in WordNet.

The annotation methodology used in (Magnini and Cavaglià, 2000) for creating WordNetDomains was mainly manual, and based on lexico-semantic criteria which take advantage from the already existing conceptual relations in WordNet. First, a small number of high level synsets were manually annotated with their correct domains. Then, an automatic procedure exploiting some of the WordNet relations (i.e. hyponymy, troponymy, meronymy, antonymy and pertain-to) was used in order to extend these assignments to all the synsets reachable through inheritance. For example, this procedure automatically marked the synset {`beak, bill, neb, nib`} with the code ZOOLOGY, starting from the fact that the synset {`bird`} was itself tagged with ZOOLOGY, and following a "part-of" relation (one of the meronymic relations present in WordNet). In some cases the inheritance procedure had to be manually blocked, inserting an "exception" in order to prevent a wrong propagation. For instance, if blocking had not been used, the term `barber_chair#1`, being a "part-of" `barbershop#1`, which is annotated with COMMERCE, would have inherited COMMERCE, which is unsuitable.

For the purpose of the experiments reported in this paper, we have used a simplified variant of WordNetDomains, called WordNetDomains(42). This was obtained from WordNetDomains by considering only 42 highly relevant labels, and tagging by a given domain c_i also the synsets that, in WordNetDomains, were tagged by the domains immediately related to c_i in a hierarchical sense (that is, the parent domain of c_i and all the children domains of c_i). For instance, the domain SPORT is retained into WordNetDomains(42), and labels both the synsets that it originally labelled in WordNetDomains, plus the ones that in WordNetDomains were labelled under its children categories (e.g. VOLLEY, BASKETBALL, ...) or under its parent category (FREE-TIME). Since FREE-TIME has another child (PLAY) which is also retained in WordNetDomains(42), the synsets originally labelled by FREE-TIME will now be labelled also by PLAY, and will thus have multiple labels. However, that a synset may have multiple labels is true in general, i.e. these labels need not have any particular relation in the hierarchy.

This restriction to the 42 most significant categories allows to obtain a good compromise between the conflicting needs of avoiding data sparseness and preventing the loss of relevant semantic information. These 42 categories belong to 5 groups, where the categories in a given group are all the children of the same WordNetDomains category, which is however not retained into WordNetDomains(42); for example, one group is formed by SPORT and PLAY, which are both children of FREE-TIME (not included into WordNetDomains(42)).

3.3.3. The experiment

We have run several experiments for different choices of the subset of RCVI chosen as corpus of text θ_y, and for different choices of the subsets of WordNetDomains(42) chosen as training set Tr_y and test set Te_y. We first describe how we have run a generic experiment, and then go on to describe the sequence of different experiments we have run. For the moment being we have run experiments consisting of one iteration only of the bootstrapping process. In future experiments we also plan to allow for multiple iterations, in which the system learns new terms also from previously learnt ones.

In our experiments we considered only nouns, thereby discarding words tagged by other syntactic categories. We plan to also consider words other than nouns in future ex-

[6]From the point of view of our term categorization task, the fact that more than one domain may be attached to the same synset means that ours is a *multi-label* categorization task (Sebastiani, 2002, Section 2.2).

periments.

For each experiment, we discarded all documents that did not contain any term from the training lexicon Tr_y, since they do not contribute in representing the meaning of training documents, and thus could not possibly be of any help in building the classifiers. Next, we discarded all "empty" training terms, i.e. training terms that were not contained in any document of θ_y, since they could not possibly contribute to learning the classifiers. Also empty test terms were discarded, since no algorithm that extracts terms from corpora could possibly extract them. Quite obviously, we also do not use the terms that occur in θ_y but belong neither to the training set Tr_y nor to the test set Te_y.

We then lemmatized all remaining documents and annotated the lemmas with part-of-speech tags, both by means of the TREETAGGER package (Schmid, 1994); we also used the WordNet morphological analyzer in order to resolve ambiguities and lemmatization mistakes. After tagging, we applied a filter in order to identify the words actually contained in WordNet, including multiwords, and then we discarded all terms but nouns. The final set of terms that resulted from this process was randomly divided into a training set Tr_y (consisting of two thirds of the entire set) and a test set Te_y (one third). As negative training examples of category c_i we chose all the training terms that are not positive examples of c_i.

Note that in this entire process we have not considered the grouping of terms into synsets; that is, the lexical units of interest in our application are the terms, and not the synsets. The reason is that RCVI is not a sense-tagged corpus, and for any term occurrence τ it is not clear to which synset τ refers to.

3.3.4. The results

Our experimental results on this task are still very preliminary, and are reported in Table 2.

Instead of tackling the entire RCVI corpus head on, for the moment being we have run only small experiments on limited subsets of it (up to 8% of its total size), with the purpose of getting a feel for which are the dimensions of the problem that need investigation; for the same reason, for the moment being we have used only a small number of boosting iterations (500). In Table 2, the first three lines concern experiments on the news stories produced on a single day (08.11.1996); the next three lines use the news stories produced in a single week (08.11.1996 to 14.11.1996), and the last six lines use the news stories produced in an entire month (01.11.1996 to 30.11.1996). Only training and test terms occurring in at least x documents were considered; the experiments reported in the same block of lines differ for the choice of the x parameter.

There are two main conclusions we can draw from these still preliminary experiments. The first conclusion is that F_1 values are still low, at least if compared to the F_1 values that have been obtained in *text* categorization research on the same corpus (Ault and Yang, 2001); a lot of work is still needed in tuning this approach in order to obtain significant categorization performance. The low values of F_1 are mostly the result of low recall values, while precision tends to be much higher, often well above the 70% mark. Note that the low absolute performance might also be explained, at least partially, with the imperfect quality of the WordNetDomains(42) resource, which was generated by a combination of automatic and manual procedures and did no undergo extensive checking afterwards.

The second conclusion is that results show a constant and definite improvement when higher values of x are used, despite the fact that higher levels of x mean a higher number of labels per term, i.e. more polysemy. This is not surprising, since when a term occurs e.g. in one document only, this means that only one entry in the vector that represents the term is non-null (i.e. significant). This is in sharp contrast with text categorization, in which the number of non-null entries in the vector representing a document equals the number of distinct terms contained in the document, and is usually at least in the hundreds. This alone might suffice to justify the difference in performance between term categorization and text categorization.

However, one reason the actual F_1 scores are low is that this is a hard task, and the evaluation standards we have adopted are considerably tough. This is discussed in the next paragraph.

No baseline? Note that we present no baseline, either published or new, against which to compare our results, for the simple fact that term categorization as we conceive it here is a novel task, and there are as yet no previous results or known approaches to the problem to compare with.

Only (Riloff and Shepherd, 1999; Roark and Charniak, 1998) have approached the problem of extending an existing thematic lexicon with new terms drawn from a text corpus. However, there are key differences between their evaluation methodology and ours, which makes comparisons difficult and unreliable. First, their "training" terms have not been chosen randomly our of a thematic dictionary, but have been carefully selected through a manual process by the authors themselves. For instance, (Riloff and Shepherd, 1999) choose words that are "frequent in the domain" and that are "(relatively) unambiguous". Of course, their approach makes the task easier, since it allows the "best" terms to be selected for training. Second, (Riloff and Shepherd, 1999; Roark and Charniak, 1998) extract the terms from texts that are known to be about the theme, which makes the task easier than ours; conversely, by using generic texts, we avoid the costly process of labelling the documents by thematic categories, and we are able to generate thematic lexicons for multiple themes at once *from the same unlabelled text corpus*. Third, their evaluation methodology is manual, i.e. subjective, in the sense that the authors themselves manually checked the results of their experiments, judging, for each returned term, how reasonable the inclusion of the term in the lexicon is[7]. This sharply contrasts with our evaluation methodology, which is completely automatic (since we measure the proficiency

[7]For instance, (Riloff and Shepherd, 1999) judged a word classified into a category correct also if they judged that "the word refers to a part of a member of the category", thereby judging the words `cartridge` and `clips` to belong to the domain WEAPONS. This looks to us a loose notion of category mambership, and anyway points to the pitfalls of "subjective" evaluation methodologies.

# of docs	# of training terms	# of test terms	# of labels per term	minimum # of docs per term	Precision micro	Recall micro	F_1 micro	Precision macro	Recall macro	F_1 macro
2,689	4,424	2,212	1.96	1	0.542029	0.043408	0.080378	0.584540	0.038108	0.071551
2,689	1,685	842	2.36	5	0.512903	0.079580	0.137782	0.487520	0.078677	0.135489
2,689	1,060	530	2.55	10	0.517544	0.086131	0.147685	0.560876	0.084176	0.146383
16,003	7,975	3,987	1.76	1	0.720165	0.049631	0.092863	0.701141	0.038971	0.073837
16,003	4,132	2,066	2.02	5	0.733491	0.075121	0.136284	0.738505	0.065472	0.120281
16,003	2,970	1,485	2.15	10	0.740260	0.091405	0.162718	0.758044	0.078162	0.141712
67,953	11,313	5,477	1.66	1	0.704251	0.043090	0.081211	0.692819	0.034241	0.065256
67,953	6,829	3,414	1.83	5	0.666667	0.040816	0.076923	0.728300	0.050903	0.095155
67,953	5,335	2,668	1.92	10	0.712406	0.076830	0.138701	0.706678	0.056913	0.105342
67,953	4,521	2,261	1.99	15	0.742574	0.086445	0.154863	0.731530	0.064038	0.117766
67,953	3,317	1,659	2.10	30	0.745455	0.098439	0.173913	0.785371	0.075573	0.137878
67,953	2,330	1,166	2.25	60	0.760417	0.117789	0.203982	0.755136	0.086809	0.155718

Table 2: Preliminary results obtained on the automated lexicon generation task (see Section 3.3. for details).

of our system at discovering terms about the theme, by the capability of the system to replicate the lexicon generation work of a lexicographer), can be replicated by other researchers, and is unaffected by possible experimenter's bias. Fourth, checking one's results for "reasonableness", as (Riloff and Shepherd, 1999; Roark and Charniak, 1998) do, means that one can only ("subjectively") measure precision (i.e. whether the terms spotted by the algorithm do in fact belong to the theme), but not recall (i.e. whether the terms belonging to the theme have actually been spotted by the algorithm). Again, this is in sharp contrast with our methodology, which ("objectively") measures precision, recall, and a combination of them. Also, note that in terms of precision, i.e. the measure that (Riloff and Shepherd, 1999; Roark and Charniak, 1998) subjectively compute, our algorithm fares pretty well, mostly scoring higher than 70% even in these very preliminary experiments.

4. Related work

4.1. Automated generation of lexical resources

The automated generation of lexicons from text corpora has a long history, dating back at the very least to the seminal works of Lesk, Salton and Sparck Jones (Lesk, 1969; Salton, 1971; Sparck Jones, 1971), and has been the subject of active research throughout the last 30 years, both within the information retrieval community (Crouch and Yang, 1992; Jing and Croft, 1994; Qiu and Frei, 1993; Ruge, 1992; Schütze and Pedersen, 1997) and the NLP community (Grefenstette, 1994; Hirschman et al., 1988; Riloff and Shepherd, 1999; Roark and Charniak, 1998; Tokunaga et al., 1995). Most of the lexicons built by these works come in the form of *cluster-based thesauri*, i.e. networks of groups of synonymous or quasi-synonymous words, in which edges connecting the nodes represent semantic contiguity. Most of these approaches follow the basic pattern of (i) measuring the degree of pairwise similarity between the words extracted from a corpus of texts, and (ii) clustering these words based on the computed similarity values. When the lexical resources being built are of a *thematic* nature, the thematic nature of a word is usually established by checking whether its frequency within thematic documents is higher than its frequency in generic documents (Chen et al., 1996; Riloff and Shepherd, 1999; Schatz et al., 1996; Sebastiani, 1999) (this property is often called *salience* (Yarowsky, 1992)).

In the approach described above, the key decision is how to tackle step (i), and there are two main approaches to this. In the first approach the similarity between two words is usually computed in terms of their degree of co-occurrence and co-absence within the same document (Crouch, 1990; Crouch and Yang, 1992; Qiu and Frei, 1993; Schäuble and Knaus, 1992; Sheridan and Ballerini, 1996; Sheridan et al., 1997); variants of this approach are obtained by restricting the context of co-occurrence from the document to the paragraph, or to the sentence (Schütze, 1992; Schütze and Pedersen, 1997), or to smaller linguistic units (Riloff and Shepherd, 1999; Roark and Charniak, 1998). In the second approach this similarity is computed from head-modifier structures, by relying on the assumption that frequent modifiers of the same word are semantically similar (Grefenstette, 1992; Ruge, 1992; Strzalkowski, 1995). The latter approach can also deal with indirect co-occurrence[8], but the former is conceptually simpler, since it does not even need any parsing step.

This literature (apart from (Riloff and Shepherd, 1999; Roark and Charniak, 1998), which are discussed below) has thus taken an *unsupervised* learning approach, which can be summarized in the recipe "from a set of documents about theme t and a set of generic documents (i.e. mostly not about t), extract the words that mostly characterize t". Our work is different, in that its underlying *supervised* learning approach requires a starting kernel of terms about t, but does not require that the corpus of documents from which

[8]We say that words w_1 and w_2 *co-occur directly* when they both occur in the same document (or other linguistic context), while we say that they *co-occur indirectly* when, for some other word w_3, w_1 and w_3 co-occur directly and w_2 and w_3 co-occur directly. Perfect synonymy is not revealed by direct co-occurrence, since users tend to consistently use either one or the other synonym but not both, while it is obviously revealed by indirect co-occurrence. However, this latter also tends to reveal many more "spurious" associations than direct co-occurrence.

the terms are extracted be labelled. This makes our supervised technique particularly suitable for *extending* a previously existing thematic lexical resource, while the previously known unsupervised techniques tend to be more useful for generating one from scratch. This suggests an interesting methodology of (i) generating a thematic lexical resource by some unsupervised technique, and then (ii) extending it by our supervised technique. An intermediate approach between these two is the one adopted in (Riloff and Shepherd, 1999; Roark and Charniak, 1998), which also requires a starting kernel of terms about t, but also requires a set of documents about theme t from which the new terms are extracted.

As anyone involved in applications of supervised machine learning knows, labelled resources are often a bottleneck for learning algorithms, since labelling items by hand is expensive. Concerning this, note that our technique is advantageous, since it requires an initial set of labelled terms *only in the first bootstrapping iteration*. Once a lexical resource has been extended with new terms, extending it further only requires a new *unlabelled* corpus of documents, but no other labelled resource. This is different from the other techniques described earlier, which require, for extending a lexical resource that has just been built by means of them, a new *labelled* corpus of documents.

A work which is closer in spirit to ours than the abovementioned ones is (Tokunaga et al., 1997), since it deals with using previously classified terms as training examples in order to classify new terms. This work exploits a naive Bayesian model for classification in conjunction with another learning method, chosen among nearest neighbour, "category-based" (by which the authors basically mean a Rocchio method – see e.g. (Sebastiani, 2002, Section 6.7)) and "cluster-based" (which does not use category labels of training examples). However, these latter learning methods and (especially) the nature of their integration with the naive Bayesian model are not specified in mathematical detail, which does not allow us to make a precise comparison between the model of (Tokunaga et al., 1997) and ours. Anyway, our model is more elegant, in that it just assumes a single learning method (for which we have chosen boosting, although we might have chosen any other supervised learning method), and in that it replaces the ad-hoc notion of "co-occurrence" with a theoretically sounder "dual" theory of text indexing, which allows one, among other things, to bring to bear any kind of intuitions on term weighting, or any kind of text indexing theory, that are known from information retrieval.

4.2. Boosting

Boosting has been applied to several learning tasks related to text analysis, including POS-tagging and PP-attachment (Abney et al., 1999), clause splitting (Carreras and Màrquez, 2001b), word segmentation (Shinnou, 2001), word sense disambiguation (Escudero et al., 2000), text categorization (Schapire and Singer, 2000; Schapire et al., 1998; Sebastiani et al., 2000; Taira and Haruno, 2001), e-mail filtering (Carreras and Márquez, 2001a), document routing (Iyer et al., 2000; Kim et al., 2000), and term extraction (Vivaldi et al., 2001). Among these works, the one somehow closest in spirit to ours is (Vivaldi et al., 2001), since it is concerned with extracting medical terms from a corpus of texts. A key difference with our work is that the features by which candidate terms are represented in (Vivaldi et al., 2001) are not simply the documents they occur in, but the results of term extraction algorithms; therefore, our approach is simpler and more general, since it does not require the existence of separate term extraction algorithms.

5. Conclusion

We have reported work in progress on the semi-automatic generation of thematic lexical resources by the combination of (i) a dual interpretation of IR-style text indexing theory and (ii) a boosting-based machine learning approach. Our method does not require pre-existing semantic knowledge, and is particularly suited to the situation in which one or more preexisting thematic lexicons need to be extended and no corpora of texts classified according to the themes are available. We have run only initial experiments, which suggest that the approach is viable, although large margins of improvement exist. In order to improve the overall performance we are planning several modifications to our currently adopted strategy.

The first modification consists in performing *feature selection*, as commonly used in text categorization (Sebastiani, 2002, Section 5.4). This will consist in individually scoring (by means of the *information gain* function) all documents in terms of how indicative they are of the occurrence or non-occurrence of the categories we are interested in, and to choose only the best-scoring ones out of a potentially huge corpus of available documents.

The second avenue we intend to follow consists in trying alternative notions of what a document is, by considering as "documents" paragraphs, or sentences, or even smaller, syntactically characterized units (as in (Riloff and Shepherd, 1999; Roark and Charniak, 1998)), rather than full-blown Reuters news stories.

A third modification consists in selecting, as the negative examples of a category c_i, all the training examples that are not positive examples of c_i *and* are at the same time positive examples of (at least one of) the siblings of c_i. This method, known as the *query-zoning method* or as the *method of quasi-positive examples*, is known to yield superior performance with respect to the method we currently use (Dumais and Chen, 2000; Ng et al., 1997).

The last avenue for improvement is the optimization of the parameters of the boosting process. The obvious parameter that needs to be optimized is the number of boosting iterations, which we have kept to a minimum in the reported experiments. A less obvious parameter is the form of the initial distribution on the training examples (that we have not described here for space limitations); by changing it with respect to the default value (the uniform distribution) we will be able to achieve a better compromise between precision and recall (Schapire et al., 1998), which for the moment being have widely different values.

Acknowledgments

We thank Henri Avancini for help with the coding task and Pio Nardiello for assistance with the

ADABOOST.MH^{KR} code. Above all, we thank Roberto Zanoli for help with the coding task and for running the experiments.

6. References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and PP attachment. In *Proceedings of EMNLP-99, 4th Conference on Empirical Methods in Natural Language Processing*, pages 38–45, College Park, MD.

Thomas Ault and Yiming Yang. 2001. kNN, Rocchio and metrics for information filtering at TREC-10. In *Proceedings of TREC-10, 10th Text Retrieval Conference*, Gaithersburg, US.

Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Amita G. Chin, editor, *Text Databases and Document Management: Theory and Practice*, pages 78–102. Idea Group Publishing, Hershey, US.

Xavier Carreras and Lluís Márquez. 2001a. Boosting trees for anti-spam email filtering. In *Proceedings of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov Chark, BG.

Xavier Carreras and Lluís Màrquez. 2001b. Boosting trees for clause splitting. In *Proceedings of CONLL-01, 5th Conference on Computational Natural Language Learning*, Toulouse, FR.

Hsinchun Chen, Chris Schuffels, and Rich Orwing. 1996. Internet categorization and search: A machine learning approach. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries*, 7(1):88–102.

Carolyn J. Crouch and Bokyung Yang. 1992. Experiments in automated statistical thesaurus construction. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 77–87, Kobenhavn, DK.

Carolyn J. Crouch. 1990. An approach to the automatic construction of global thesauri. *Information Processing and Management*, 26(5):629–640.

Susan T. Dumais and Hao Chen. 2000. Hierarchical classification of Web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR. ACM Press, New York, US.

Gerard Escudero, Lluís Màrquez, and German Rigau. 2000. Boosting applied to word sense disambiguation. In *Proceedings of ECML-00, 11th European Conference on Machine Learning*, pages 129–141, Barcelona, ES.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, US.

Gregory Grefenstette. 1992. Use of syntactic context to produce term association lists for retrieval. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 89–98, Kobenhavn, DK.

Gregory Grefenstette. 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers, Dordrecht, NL.

Lynette Hirschman, Ralph Grishman, and Naomi Sager. 1988. Grammatically-based automatic word class formation. *Information Processing and Management*, 11(1/2):39–57.

Raj D. Iyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal. 2000. Boosting for document routing. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 70–77, McLean, US.

Yufeng Jing and W. Bruce Croft. 1994. An association thesaurus for information retrieval. In *Proceedings of RIAO-94, 4th International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 146–160, New York, US.

Kyo Kageura and Bin Umino. 1996. Methods of automatic term recognition: a review. *Terminology*, 3(2):259–289.

Yu-Hwan Kim, Shang-Yoon Hahn, and Byoung-Tak Zhang. 2000. Text filtering by boosting naive Bayes classifiers. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 168–75, Athens, GR.

Alberto Lavelli, Bernardo Magnini, and Fabrizio Sebastiani. 2002. Building thematic lexical resources by term categorization. Technical report, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT. Forthcoming.

Michael E. Lesk. 1969. Word-word association in document retrieval systems. *American Documentation*, 20(1):27–38.

David D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50, Kobenhavn, DK.

Bernardo Magnini and Gabriela Cavaglià. 2000. Integrating subject field codes into WordNet. In *Proceedings of LREC-2000, 2nd International Conference on Language Resources and Evaluation*, pages 1413–1418, Athens, GR.

Lois Mai Chan, John P. Comaromi, Joan S. Mitchell, and Mohinder Satija. 1996. *Dewey Decimal Classification: a practical guide*. OCLC Forest Press, Albany, US, 2nd edition.

Hwee T. Ng, Wei B. Goh, and Kok L. Low. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, pages 67–73, Philadelphia, US. ACM Press, New York, US.

Helen J. Peat and Peter Willett. 1991. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5):378–383.

Paul Procter, editor. 1978. *The Longman Dictionary of Contemporary English*. Longman, Harlow, UK.

Yonggang Qiu and Hans-Peter Frei. 1993. Concept-based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Devel-*

opment in Information Retrieval, pages 160–169, Pittsburgh, US.

Ellen Riloff and Jessica Shepherd. 1999. A corpus-based bootstrapping algorithm for semi-automated semantic lexicon construction. *Journal of Natural Language Engineering*, 5(2):147–156.

Brian Roark and Eugene Charniak. 1998. Noun phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of ACL-98, 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116, Montreal, CA.

Gerda Ruge. 1992. Experiments on linguistically-based terms associations. *Information Processing and Management*, 28(3):317–332.

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

Gerard Salton and Michael J. McGill. 1983. *Introduction to modern information retrieval*. McGraw Hill, New York, US.

Gerard Salton. 1971. Experiments in automatic thesaurus construction for information retrieval. In *Proceedings of the IFIP Congress*, volume TA-2, pages 43–49, Ljubljana, YU.

Robert E. Schapire and Yoram Singer. 2000. BOOSTEXTER: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

Robert E. Schapire, Yoram Singer, and Amit Singhal. 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, AU.

Bruce R. Schatz, Eric H. Johnson, Pauline A. Cochrane, and Hsinchun Chen. 1996. Interactive term suggestion for users of digital libraries: Using subject thesauri and co-occurrence lists for information retrieval. In *Proceedings of DL-96, 1st ACM Digital Library Conference*, pages 126–133, Bethesda, US.

Peter Schäuble and Daniel Knaus. 1992. The various roles of information structures. In *Proceedings of the 16th Annual Conference of the Gesellschaft für Klassifikation*, pages 282–290, Dortmund, DE.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK.

Hinrich Schütze and Jan O. Pedersen. 1997. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3):307–318.

Hinrich Schütze. 1992. Dimensions of meaning. In *Proceedings of Supercomputing'92*, pages 787–796, Minneapolis, US.

Fabrizio Sebastiani, Alessandro Sperduti, and Nicola Valdambrini. 2000. An improved boosting algorithm and its application to automated text categorization. In *Proceedings of CIKM-00, 9th ACM International Conference on Information and Knowledge Management*, pages 78–85, McLean, US.

Fabrizio Sebastiani. 1999. Automated generation of category-specific thesauri for interactive query expansion. In *Proceedings of IDC-99, 9th International Database Conference on Heterogeneous and Internet Databases*, pages 429–432, Hong Kong, CN.

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

Páraic Sheridan and Jean-Paul Ballerini. 1996. Experiments in multilingual information retrieval using the SPIDER system. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 58–65, Zürich, CH.

Páraic Sheridan, Martin Braschler, and Peter Schäuble. 1997. Cross-language information retrieval in a multilingual legal domain. In *Proceedings of ECDL-97, 1st European Conference on Research and Advanced Technology for Digital Libraries*, pages 253–268, Pisa, IT.

Hiroyuki Shinnou. 2001. Detection of errors in training data by using a decision list and AdaBoost. In *Proceedings of the IJCAI-01 Workshop on Text Learning: Beyond Supervision*, Seattle, US.

Karen Sparck Jones. 1971. *Automatic keyword classification for information retrieval*. Butterworths, London, UK.

Tomek Strzalkowski. 1995. Natural language information retrieval. *Information Processing and Management*, 31(3):397–417.

Hirotoshi Taira and Masahiko Haruno. 2001. Text categorization using transductive boosting. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*, pages 454–465, Freiburg, DE.

Takenobu Tokunaga, Makoto Iwayama, and Hozumi Tanaka. 1995. Automatic thesaurus construction based on grammatical relations. In *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, pages 1308–1313, Montreal, CA.

Takenobu Tokunaga, Atsushi Fujii, Makoto Iwayama, Naoyuki Sakurai, and Hozumi Tanaka. 1997. Extending a thesaurus by classifying words. In *Proceedings of the ACL-EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources*, pages 16–21, Madrid, ES.

Jordi Vivaldi, Lluís Màrquez, and Horacio Rodríguez. 2001. Improving term extraction by system combination using boosting. In *Proceedings of ECML-01, 12th European Conference on Machine Learning*, pages 515–526, Freiburg, DE.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of COLING-92, 14th International Conference on Computational Linguistics*, pages 454–460, Nantes, FR.

Learning Grammars for Noun Phrase Extraction by Partition Search

Anja Belz

ITRI
University of Brighton
Lewes Road
Brighton BN2 4GJ, UK
Anja.Belz@itri.brighton.ac.uk

Abstract

This paper describes an application of Grammar Learning by Partition Search to noun phrase extraction, an essential task in information extraction and many other NLP applications. Grammar Learning by Partition Search is a general method for automatically constructing grammars for a range of parsing tasks; it constructs an optimised probabilistic context-free grammar by searching a space of nonterminal set partitions, looking for a partition that maximises parsing performance and minimises grammar size. The idea is that the considerable time and cost involved in building new grammars can be avoided if instead existing grammars can be automatically adapted to new parsing tasks and new domains. This paper presents results for applying Partition Search to the tasks of (i) identifying flat NP chunks, and (ii) identifying all NPs in a text. For NP chunking, Partition Search improves a general baseline result by 12.7%, and a method-specific baseline by 2.2%. For NP identification, Partition Search improves the general baseline by 21.45%, and the method-specific one by 3.48%. Even though the grammars are nonlexicalised, results for NP identification closely match the best existing results for lexicalised approaches.

1. Introduction

Grammar Learning by Partition Search is a computational learning method that constructs probabilistic grammars optimised for a given parsing task. Its main practical application is the adaptation of grammars to new tasks, in particular the adaptation of conventional, "deep" grammars to the shallow parsing tasks involved in many NLP applications. The parsing tasks investigated in this paper are NP identification and NP chunking both of which involve the detection of NP boundaries, a task which is fundamental to information extraction and retrieval, text summarisation, document classification, and other applications.

The ability to automatically adapt an existing grammar to a new parsing task saves time and expense. Furthermore, adapting deep grammars to shallow parsing tasks has a specific advantage. Existing approaches to NP extraction are mostly completely flat. They do not carry out any structural analysis above the level of the chunks and phrases they are meant to detect. Using Partition Search to adapt deep grammars for shallow parsing permits those parts of deeper structural analysis to be retained that are useful for the detection of more shallow components.

The remainder of this paper is organised in two main sections. Section 2. describes Grammar Learning by Partition Search. Section 3. reports experiments and results for NP identification and NP chunking.

2. Learning PCFGs by Partition Search

Partition Search Grammar Learning starts from the idea that new context-free grammars can be created from old simply by modifying the nonterminal sets, *merging* and *splitting* subsets of nonterminals. For example, for certain parsing tasks it is useful to *split* a single verb phrase category into verb phrases that are headed by a modal verb and those that are not, whereas for other parsing tasks, the added grammar complexity is avoidable. In another context, it may not be necessary to distinguish noun phrases in subject position from first objects and second objects, making it possible to *merge* the three categories into one.

The usefulness of such split and merge operations can be objectively measured by their effect on a grammar's size (number of rules and nonterminals) and performance (parsing accuracy on a given task). Grammar Learning by Partition Search automatically tries out different combinations of merge and split operations and therefore can automatically optimise a grammar's size and performance.

2.1. Preliminary definitions

Definition 1 Set Partition

A partition of a nonempty set A is a subset Π of 2^A such that $\emptyset$ is not an element of Π and each element of A is in one and only one set in Π.

The partition of A where all elements are singleton sets is called the *trivial partition* of A.

Definition 2 Probabilistic Context-Free Grammar

A Probabilistic Context-Free Grammar (PCFG) is a 4-tuple (W, N, N^S, R), where W is a set of terminal symbols, N is a set of nonterminal symbols, $N^S = \{(s_1, p(s_1)), \dots (s_l, p(s_l))\}, \{s_1, \dots s_l\} \subseteq N$ is a set of start symbols with associated probabilities summing to one, and $R = \{(r_1, p(r_1)), \dots (r_m, p(r_m))\}$ is a set of rules with associated probabilities. Each rule r_i is of the form $n \rightarrow \alpha$, where n is a nonterminal, and α is a string of terminals and nonterminals. For each nonterminal n, the values of all $p(n \rightarrow \alpha_i)$ sum to one, or: $\sum_{i:(n \rightarrow \alpha_i, p(n \rightarrow \alpha_i)) \in R} p(n \rightarrow \alpha_i) = 1$.

2.2. Generalising and Specialising PCFGs through Nonterminal Set Operations

2.2.1. Nonterminal merging

Consider two PCFGs G and G':

$G = (W, N, N^S, R)$,
  $W =$ { NNS, DET, NN, VBD, JJ }
  $N =$ { S, NP-SUBJ, VP, NP-OBJ }
  $N^S =$ { (S, 1) }
  $R =$ { (S -> NP-SUBJ VP, 1),
    (NP-SUBJ -> NNS, 0.5),
    (NP-SUBJ -> DET NN, 0.5),
    (VP -> VBD NP-OBJ, 1),
    (NP-OBJ -> NNS, 0.75),
    (NP-OBJ -> DET JJ NNS, 0.25) }

$G' = (W, N', N^S, R')$,
  $W =$ { NNS, DET, NN, VBD, JJ }
  $N' =$ { S, NP, VP }
  $N^S =$ { (S, 1) }
  $R' =$ { (S -> NP VP, 1),
    (NP -> NNS, 0.625),
    (NP -> DET NN, 0.25),
    (VP -> VBD NP, 1),
    (NP -> DET JJ NNS, 0.125) }

Intuitively, to derive G' from G, the two nonterminals `NP-SUBJ` and `NP-OBJ` are merged into a single new nonterminal `NP`. This merge results in two rules from R becoming identical in R': both `NP-SUBJ -> NNS` and `NP-OBJ -> NNS` become `NP -> NNS`. One way of determining the probability of the new rule `NP -> NNS` is to sum the probabilities of the old rules and renormalise by the number of nonterminals that are being merged[1]. In the above example therefore $p(\texttt{NP -> NNS}) = (0.5 + 0.75)/2 = 0.625$[2].

An alternative would be to reestimate the new grammar on some corpus, but this is not appropriate in the current context: merge operations are used in a search process (see below), and it would be expensive to reestimate each new candidate grammar derived by a merge. It is better to use any available training data to estimate the original grammar's probabilities, then the probabilities of all derived grammars can simply be calculated as described above without expensive corpus reestimation.

The new grammar G' derived from an old grammar G by merging nonterminals in G is a generalisation of G: the language of G', or $L(G')$, is a superset of the language of G, or $L(G)$. E.g., `det jj nns vbd det jj nns` is in $L(G')$ but not in $L(G)$. The set of parses assigned to a sentence s by G' differs from the set of parses assigned to s by G. The probabilities of parses for s can change, and so can the probability ranking of the parses, i.e. the most likely parse for s under G may be different from the most likely parse for s under G'. Finally, G' has the same number of rules as G or fewer.

[1]Reestimating the probabilities on the training corpus would of course produce identical results.

[2]Renormalisation is necessary because the probabilities of all rules expanding the same nonterminal sum to one, therefore the probabilities of all rules expanding a new nonterminal resulting from merging n old nonterminals will sum to n.

2.2.2. Nonterminal splitting

Deriving a new PCFG from an old one by splitting nonterminals in the old PCFG is not quite the exact reverse of deriving a new PCFG by merging nonterminals. The difference lies in determining probabilities for new rules. Consider the following grammars G and G':

$G = (W, N, N^S, R)$,
  $W =$ { NNS, DET, NN, VBD, JJ }
  $N =$ { S, NP, VP }
  $N^S =$ { (S, 1) }
  $R =$ { (S -> NP VP, 1),
    (NP -> NNS, 0.625),
    (NP -> DET NN, 0.25),
    (VP -> VBD NP, 1),
    (NP -> DET JJ NNS, 0.125) }

$G' = (W, N', N^S, R')$,
  $W =$ { NNS, DET, NN, VBD, JJ }
  $N' =$ { S, NP-SUBJ, VP, NP-OBJ }
  $N^S =$ { (S, 1) }
  $R' =$ { (S -> NP-SUBJ VP, ?),
    (S -> NP-OBJ VP, ?),
    (NP-SUBJ -> NNS, ?),
    (NP-SUBJ -> DET NN, ?),
    (NP-SUBJ -> DET JJ NNS, ?) }
    (VP -> VBD NP-SUBJ, ?),
    (VP -> VBD NP-OBJ, ?),
    (NP-OBJ -> NNS, ?),
    (NP-OBJ -> DET NN, ?),
    (NP-OBJ -> DET JJ NNS, ?) }

To derive G' from G, the single nonterminal `NP` is split into two nonterminals `NP-SUBJ` and `NP-OBJ`. This split results in several new rules. For example, for the old rule `NP -> NNS`, there now are two new rules `NP-SUBJ -> NNS` and `NP-OBJ -> NNS`. One possibility for determining the new rule probabilities is to redistribute the old probability mass evenly among them, i.e. $p(\texttt{NP -> NNS}) = p(\texttt{NP-SUBJ -> NNS}) = p(\texttt{NP-SUBJ -> NNS})$. However, then there would be no benefit at all from performing such a split: the resulting grammar would be larger, the most likely parses remain unchanged, and for each parse p under G that contains a nonterminal participating in a split operation, there would be at least two equally likely parses under G'.

The new probabilities cannot be calculated directly from G. The redistribution of the probability mass has to be motivated from a knowledge source outside of G. One way to proceed is to estimate the new rule probabilities on the original corpus — provided that it contains the information on the basis of which a split operation was performed in extractable form. For the current example, a corpus in which objects and subjects are annotated could be used to estimate the probabilities of the rules in G', and might yield the following result (which reflects the fact that in English, the `NP` in a sentence `NP VP` is usually a subject, whereas the `NP` in a `VP` consisting of a verb followed by an `NP` is an object):

$$
\begin{array}{rll}
G' = (W, N', N^S, R'), & & \\
W = & \{ \texttt{NNS, DET, NN, VBD, JJ} \} & \\
N' = & \{ \texttt{S, NP-SUBJ, VP, NP-OBJ} \} & \\
N^S = & \{ (\texttt{S}, 1) \} & \\
R' = & \{ & (\texttt{S -> NP-SUBJ VP}, 1), \\
 & & (\texttt{S -> NP-OBJ VP}, 0), \\
 & & (\texttt{NP-SUBJ -> NNS}, 0.5), \\
 & & (\texttt{NP-SUBJ -> DET NN}, 0.5), \\
 & & (\texttt{NP-SUBJ -> DET JJ NNS}, 0) \} \\
 & & (\texttt{VP -> VBD NP-SUBJ}, 0), \\
 & & (\texttt{VP -> VBD NP-OBJ}, 1), \\
 & & (\texttt{NP-OBJ -> NNS}, 0.75), \\
 & & (\texttt{NP-OBJ -> DET NN}, 0), \\
 & & (\texttt{NP-OBJ -> DET JJ NNS}, 0.25) \}
\end{array}
$$

With rules of zero probability removed, G' is identical to the original grammar G in the example in the previous section.

2.3. Partition Search

A PCFG together with nonterminal merge and split operations defines a space of derived grammars which can be searched for a new PCFG that optimises some given objective function. The disadvantage of this search space is that it is infinite, and each split operation requires the reestimation of rule probabilities from a training corpus, making it computationally much more expensive than a merge operation.

However, there is a simple way to make the search space finite, and at the same time to make split operations redundant. The resulting method, Grammar Learning by Partition Search, is summarised in this section (Partition Search is described in more detail, including formal definitions and algorithmic details, in Belz (2002)).

2.3.1. PCFG Partitioning

An arbitrary number of merges can be represented by a partition of the set of nonterminals. For the example presented in Section 2.2.1. above, the partition of the nonterminal set N in G that corresponds to the nonterminal set N' in G' is { {S}, {NP-SBJ, NP-OBJ}, {VP} }. The original grammar G together with a partition of its nonterminal set fully specifies the new grammar G': the new rules and probabilities, and the entire new grammar G' can be derived from the partition together with the original grammar G. The process of obtaining a new grammar G', given a base grammar G and a partition of the nonterminal set N of G will be called PCFG Partitioning[3].

2.3.2. Search space

The search space for Grammar Learning by Partition Search can be made finite and searchable entirely by merge operations (grammar partitions).

Making the search space finite: The number of merge operations that can be applied to a nonterminal set is finite, because after some finite number of merges there remains only one nonterminal. On the other hand, the number of split operations that can sensibly be applied to a nonterminal *NT* has an upper bound in the number of different terminals strings dominated by *NT* in a corpus of evidence (e.g. the corpus the PCFG was trained on). For example, when splitting the nonterminal NP into subjects and objects, there would be no point in creating more new nonterminals than the number of different subjects and objects found in the corpus.

Given these (generous) bounds, there is a finite number of distinct grammars derivable from the original grammar by different combinations of merge and split operations. This forms the basic space of candidate solutions for Grammar Learning by Partition Search.

Making the search space searchable by grammar partitioning only: Imposing an upper limit on the number and kind of split operations permitted not only makes the search space finite but also makes it possible to directly derive this *maximally split nonterminal set* (Max Set). Once the Max Set has been defined, the single grammar corresponding to it — the *maximally split Grammar* (Max Grammar) — can be derived and retrained on the training corpus.

The set of points in the search space corresponds to the set of partitions of the Max Set. Search for an optimal grammar can thus be carried out directly in the partition space of the Max Grammar.

Structuring the search space: The finite search space can be given hierarchical structure as shown in Figure 1 for an example of a very simple base nonterminal set {NP, VP, PP}, and a corpus which contains three different NPs, three different VPs and two different PPs.

At the top of the graph is the Max Set. The sets at the next level down (level 7) are created by merging pairs of nonterminals in the Max Set, and so on for subsequent levels. At the bottom is the *maximally merged nonterminal set* (Min Set) consisting of a single nonterminal *NT*. The sets at the level immediately above it can be created by splitting *NT* in different ways. The sets at level 2 are created from those at level 1 by splitting one of their elements. The original nonterminal set ends up somewhere in between the top and bottom (at level 3 in this example).

While this search space definition results in a finite search space and obviates the need for the expensive split operation, the space will still be vast for all but trivial corpora. In Section 3.3. below, alternative ways for defining the Max Set are described that result in much smaller search spaces.

2.3.3. Search task and evaluation function

The input to the Partition Search procedure consists of a base grammar G_0, a base training corpus C, and a task-specific training corpus D^T. G_0 and C are used to create the Max Grammar G. The **search task** can then be defined as follows:

> Given the maximally split PCFG $G = (W, N, N^S, R)$, a data set of sentences D, and a set of target parses D^T for D, find a partition Π_N of N that derives a grammar $G' = (W, \Pi_N, N^{S'}, R')$, such that $|R'|$ is minimised, and $f(G', D, D^T)$ is maximised, where f scores the performance of G' on D as compared to D^T.

[3]The concept of context-free grammar partitioning in this paper is not directly related to that in (Korenjak, 1969; Weng and Stolcke, 1995), and later publications by Weng et al. In these previous approaches, a non-probabilistic CFG's *set of rules* is partitioned into subsets of rules. The partition is drawn along a specific nonterminal NT, which serves as an interface through which the subsets of rules (hence, subgrammars) can communicate after partition (one grammar calling the other).

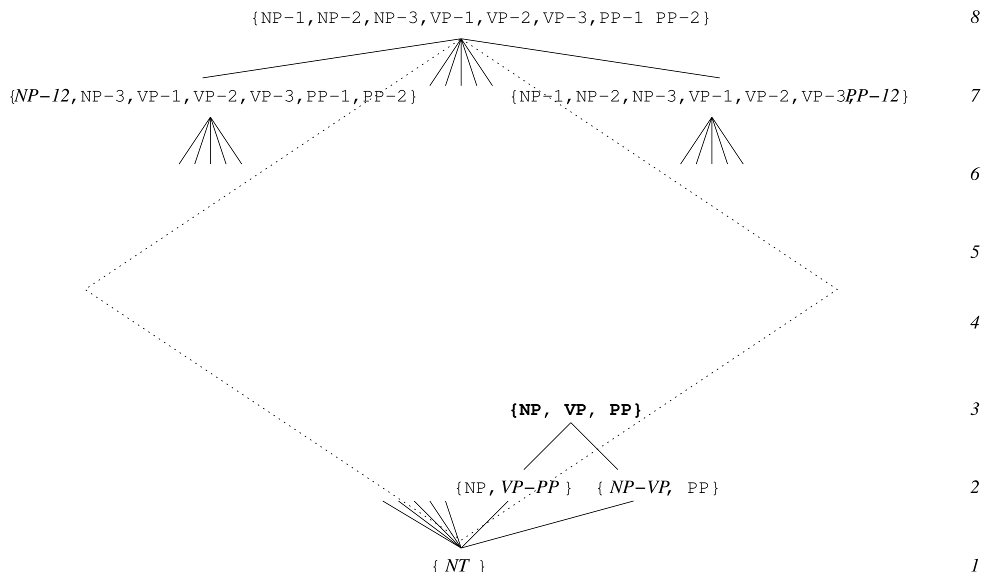

{NP-1,NP-2,NP-3,VP-1,VP-2,VP-3,PP-1 PP-2}
{NP-12,NP-3,VP-1,VP-2,VP-3,PP-1,PP-2}
{NP-1,NP-2,NP-3,VP-1,VP-2,VP-3,PP-12}
{NP, VP, PP}
{NP, VP-PP }
{ NP-VP, PP}
{ NT }
8
7
6
5
4
3
2
1

Figure 1: Simple example of a partition search space.

The size of the nonterminal set and hence of the grammar decreases from the top to the bottom of the search space. Therefore, if the partition space is searched top-down, grammar size is minimised automatically and does not need to be assessed explicitly.

In the current implementation, the **evaluation function** f simply calculates the F-Score achieved by a candidate grammar on D as compared to D^T. The F-Score is obtained by combining the standard PARSEVAL evaluation metrics *Precision* and *Recall*[4] as follows: $2 \times Precision \times Recall/(Precision + Recall)$.

An existing parser[5] was used to obtain Viterbi parses. If the parser failed to find a complete parse for a sentence, a simple grammar extension method was used to obtain partial parses instead (based on Schmid and Schulte im Walde (2000, p. 728)).

2.3.4. Search algorithm

Since each point in the search space can be accessed directly by applying the corresponding nonterminal set partition to the Max Grammar, the search space can be searched in any direction by any search method using partitions to represent candidate grammars.

In the current implementation, a variant of beam search is used to search the partition space top down. A list of the n current best candidate partitions is maintained (initialised to the Max Set). For each of the n current best partitions a random subset of size b of its children in the hierarchy is generated and evaluated. From the union of current best partitions and the newly generated candidate partitions, the n best elements are selected and form the new current best set. This process is iterated until either no new partitions can be generated that are better than their parents, or the lowest level of the partition tree is reached. In each iteration the size of the nonterminal set (partition) decreases by one.

The size of the search space grows exponentially with the size i of the Max Set. However, the complexity of the Partition Search algorithm is only $O(nbi)$, because only up to $n \times b$ partitions are evaluated in each of up to i iterations[6].

3. Learning NP Extraction Grammars

3.1. Data and Parsing Tasks

Sections 15–18 of WSJC were used for deriving the base grammar and as the base training corpus, and different randomly selected subsets of Section 1 from the same corpus were used as task-specific training corpora during search. Section 20 was used for final performance tests.

Results are reported in this paper for the following two parsing tasks. In **NP identification** the task is to identify in the input sentence all noun phrases[7], nested and otherwise, that are given in the corresponding WSJC parse. **NP chunking** was first defined by (Abney, 1991), and involves the identification of flat noun phrase chunks. Target parses were derived from WSJC parses by an existing conversion procedure[8].

The Brill Tagger was used for POS tagging testing data, and achieved an average accuracy of 97.5% (as evaluated by `evalb`).

3.2. Base grammar

A simple treebank grammar[9] was derived from Sections 15–18 of the WSJ corpus by the following procedure:

1. Iteratively edit the corpus by deleting (i) brackets and labels that correspond to empty category expansions; (ii) brackets

[4]I used the `evalb` program by Sekine and Collins (`http://cs.nyu.edu/cs/projects/proteus/evalb/`) to obtain Precision and Recall figures.

[5]LoPar (Schmid, 2000) in its non-head-lexicalised mode. Available from `http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/LoPar-en.html`.

[6]As before, n is the number of current best candidate solutions, b is the width of the beam, and i is the size of the Max Set.

[7]Corresponding to the WSJC categories `NP`, `NX`, `WHNP` and `NAC`.

[8]Devised by Erik Tjong Kim Sang for the TMR project *Learning Computational Grammars*.

[9]The term was coined by Charniak (1996).

and labels containing a single constituent that is not labelled with a POS-tag; (iii) cross-indexation tags; (iv) brackets that become empty through a deletion.

2. Convert each remaining bracketting in the corpus into the corresponding production rule.
3. Collect sets of terminals W, nonterminals N and start symbols N^S from the corpus. Probabilities p for rules $n \rightarrow \alpha$ are calculated from the rule frequencies C by Maximum Likelihood Estimation: $p(n \rightarrow \alpha) = \frac{C(n\rightarrow\alpha)}{\sum_i C(n\rightarrow\alpha^i)}$.

This procedure creates the base grammar *BARE* which has $10,118$ rules and 147 nonterminals.

3.3. Restricting the search space further

The simple method described in Section 2.3.2. for defining the maximally split nonterminal set (Max Set) tends to result in vast search spaces. Using parent node (PN) information to create the Max Set is much more restrictive and linguistically motivated. The Max Grammar *PN* used in the experiments reported below can be seen as making use of *Local Structural Context* (Belz, 2001): the independence assumptions inherent in PCFGs are weakened by making the rules' expansion probabilities dependent on part of their immediate structural context (here, its parent node). To obtain the grammar *PN*, the base grammar's nonterminal set is maximally split on the basis of the *parent node* under which rules are found in the base training corpus[10]. Several previous investigations have demonstrated improvement in parsing results due to the inclusion of parent node information (Charniak and Carroll, 1994; Johnson, 1998; Verdú-Mas et al., 2000).

Another possibility is to use the base grammar *BARE* itself as the Max Grammar. This is a very restrictive search space definition and amounts to an attempt to optimise the base grammar in terms of its size and its performance on a given task without adding any information. Results are given below for both *BARE* and *PN* as Max Grammars.

In the current implementation of the algorithm, the search space is reduced further by avoiding duplicate partitions, and by only allowing merges of nonterminals that have the same phrase prefix `NP-*, VP-*` etc.

The Max Grammars end up having sets of nonterminals that differ from the bracket labels used in the WSJC: while the phrase categories (e.g. NP) are the same, the tags (e.g. `*-S, *-3`) on the phrase category labels may differ. In the evaluation, all labels starting with the same phrase category prefix are considered equivalent.

3.4. NP chunking results

Baseline Results. Base grammar *BARE* (see Section 3.2. achieves an F-Score of 88.25 on the NP chunking task. This baseline result compares as follows with existing results:

	NP chunking
Chunk Tag Baseline	79.99
Grammar *BARE*	88.25
Current Best: nonlexicalised	90.12
lexicalised	93.25 (93.86)

[10]The parent node of a phrase is the category of the phrase that immediately contains it.

The chunk tag baseline F-Score is the standard baseline for the NP chunking task and is obtained by tagging each POS tag in a sentence with the label of the phrase that it most frequently appears in, and converting these phrase tags into labelled bracketings (Nerbonne et al., 2001, p. 102). The best nonlexicalised result was achieved with the decision-tree learner C5.0 (Tjong Kim Sang et al., 2000), and the current overall best result for NP chunking is for memory-based learning and a lexicalised chunker (Tjong Kim Sang et al., 2000)[11].

Table 1 shows results for Partition Search applied to the NP chunking task. The first column shows the Max Grammar used in a given batch of experiments. The second column indicates the type of result, where the Max Grammar result is the F-Score, grammar size and number of nonterminals of the Max Grammar itself, and the remaining results are the average and single best results achieved by Partition Search. The third and fourth columns show the number of iterations and evaluations carried out before search stopped. Columns 5–8 show details of the final solution grammars: column 5 shows the evaluation score on the training data, column 6 the overall F-Score on the testing data, column 7 the size, and the last column gives the number of nonterminals.

The best result (boldface) was an F-Score of 90.24% (compared to the base result of 88.25%), and 95 nonterminals (147 in the base grammar), while the number of rules increased from 10,118 to 11,972. This result improves the general baseline by 12.7% and the performance by grammar *BARE* by 2.2%. It also outperforms the best existing result of 90.12% for nonlexicalised NP chunking by a small margin.

3.5. NP identification results

Baseline Results. Base grammar *BARE* achieves an F-Score of 79.29 on the NP identification task. This baseline result compares as follows with existing results:

	NP identification
Chunk Tag Baseline	67.56
Grammar *BARE*	79.29
Current Best: nonlexicalised	80.15
lexicalised	83.79

All results in this table (except for that for grammar *BARE*) are reported in Nerbonne et al. (2001, p. 103). The task definition used there was slightly different in that it omitted two minor NP categories (WSJC brackets labelled NAC and NX). The slightly different task definition has only a very small effect on F-Scores, so the above results are comparable. The chunk tag baseline F-Score was again obtained by tagging each POS tag in a sentence with the label of the phrase that it most frequently appears in. The best lexicalised result was achieved with a cascade of memory-based learners. The same paper also included two results for nonlexicalised NP identification.

Table 2 (same format as Table 1) contains results for Partition Search and the NP identification task. The smallest nonterminal set had 63 nonterminals (147 in the base

[11]Nerbonne et al. (2001) report a slightly better result of 93.86 achieved by combining seven different learning systems.

Max Grammar		Iter.	Eval.	F-Score (subset)	F-Score (WSJC S 1)	Size (rules)	Nonterms
BARE	Max Grammar result:				88.25	10,118	147
	Average:	116.8	2,749.6	89.64	88.57	7,849.6	32.2
	Best (size):	119	2,806	89.79	88.51	7,541	30
	Best (F-score):	114	2,674	87.93	88.70	7,777	35
PN	Max Grammar result:				89.86	16,480	970
	Average:	526	13,007.75	94.85	89.83	14,538.25	446
	Best (size and F-score):	**877**	**21,822**	**93.85**	**90.24**	**11,972**	**95**

Table 1: Partition tree search results for NP chunking task, WSJC Section 1 (averaged over 5 runs, variable parameters: $x = 50, b = 5, n = 5$).

Max Grammar		Iter.	Eval.	F-Score (subset)	F-Score (WSJC S 1)	Size (rules)	Nonterms
BARE	Max Grammar result:				79.29	10,118	147
	Average	111.4	2,629	87.831	79.10	8,655	37.6
	Best (size):	113	2,679	86.144	78.9	8,374	36
	Best (F-score):	114	2,694	90.246	79.51	8,541	41
PN	Max Grammar result:				82.01	16,480	970
	Average:	852.6	21,051	91.2098	81.41308	13,202.8	119.4
	Best (size):	909	22,474	91.881	80.9830	12,513	63
	Best (F-score):	**658**	**16,286**	**89.572**	**82.0503**	**15,305**	**314**

Table 2: Partition tree search results for NP identification task, WSJC Section 1 (averaged over 5 runs, variable parameters: $x = 50, b = 5, n = 5$).

grammar). The best result (boldface) was an F-Score of 82.05% (base result was 79.29%), while the number of rules increased from 10,118 to 15,305. This improves the general baseline by 21.45% and grammar *BARE* by 3.48%. It also outperforms the other two results for nonlexicalised NP chunking by a significant margin, and even comes close to the best lexicalised result (83.79%).

3.6. General comments

Partition Search is able to reduce grammar size by merging groups of nonterminals (hence groups of rules) that do not need to be distinguished for a given task. It is able to improve parsing performance firstly by grammar generalisation (partitioned grammars parse a superset of the sentences parsed by the base grammar), and secondly by reranking parse probabilities (the most likely parse for a sentence under a partitioned grammar can differ from its most likely parse under the base grammar).

The margins of improvement over baseline results were bigger for the NP identification task than for NP chunking. The results reported here for NP chunking are no match for the best lexicalised results, whereas the results for NP identification come close to the best lexicalised results. This indicates that the two characteristics that most distinguish the grammars used here from other approaches — some non-shallow structural analysis and parent node information — are more helpful for NP identification.

Preliminary tests revealed that results were surprisingly constant over different combinations of variable parameter values, although training subset size of less then 50 meant unpredictable results for the complete WSJC Section 1. For a random subset of size 50 and above, there is an almost complete correspondence between subset F-Score and Section 1 F-Score, i.e. higher subset F-Score almost always means higher Section 1 F-Score.

The results presented in the previous section also show what happens if Partition Search is used as a grammar compression method (when existing grammars are used as Max Grammars). In Table 1, for example, when applied to the base grammar *BARE* (four top rows), it maximally reduces the number of nonterminals from 147 to 30 and the number of rules from $10,118$ to $7,541$, while *improving* the overall F-Score. The size reductions on the *PN* grammar are even bigger: 970 nonterminals down to 95, and $16,480$ rules down to $11,972$, again with a slight improvement in the F-Score (even though on average, the F-Score remained about the same). Unlike other grammar compression methods (Charniak, 1996; Krotov et al., 2000), Partition Search achieves lossless compression, in the sense that the compressed grammars are guaranteed to be able to parse all of the sentences parsed by the original grammar.

Compared to other approaches using parent node information (Charniak and Carroll, 1994; Johnson, 1998; Verdú-Mas et al., 2000), the approach presented here has the advantage of being able to select a subset of all parent node information on the basis of its usefulness for a given parsing task. This saves on grammar complexity, hence parsing cost.

3.7. Nonterminal distinctions preserved/eliminated

The base grammar *BARE* has 26 different phrase category prefixes (S, NP, etc.). The additional tags encoding grammatical function and parent node information results in much larger numbers of nonterminals. One of the aims of partition search is to reduce this number, preserving only useful distinctions. This section looks at nonterminal distinctions that were preserved and eliminated for each task and grammar.

3.7.1. Base grammar *BARE* (functional tags only)

Twelve of the 26 phrase categories are not annotated with functional tags in the WSJC. The remaining 14 phrase categories have between 2 and 28 grammatical function subcategories[12].

In the *BARE* grammar, more nonterminals were merged on average in the NP chunking task (32.2 remaining) than in the NP identification task (37.6 remaining). This is as might be expected since the NP identification task looks the more complex.

Results for NP chunking show a very strong tendency to merge the subcategories of all phrase categories except for two: NP and PP. With only the rare exception, the distinction between different grammatical functions is eliminated for the other 12 out of 14 phrase categories. By contrast, for NP, between 2 and 5 different categories remain (average 2.8), and for PP, between 2 and 4 remain (average 3.6). This implies that for NP chunking only the different grammatical functions of NPs and PPs are useful.

Results for NP identification show a tendency to perserve distinctions among the subcategories of SBAR, NP and PP and to a lesser extent among those of ADVP and ADJP. Other distinctions tend to be eliminated. All subcategories of SBARQ, NX, NAC, INTJ and FRAG are always merged, UCP and SINV nearly always.

3.7.2. Grammar *PN* (parent node tags)

The *PN* grammar has 970 phrase subcategories for the 26 basic phrase categories of which only those with the largest numbers of subcategories are examined here: NP (173), PP (173), ADVP (118), S (76), and VP (62).

Surprisingly, far fewer nonterminals were merged on average in the NP chunking task (446 remaining) than in the NP identification task (only 119.4 remaining).

In both tasks, although more so in the NP chunking task, the strongest tendency was that far more NP subcategories were preserved than any other.

In the NP identification task, the different NAC and NX subcategories were always merged into a single one, whereas in the NP chunking task, at least 4 different NAC and 3 different NX subcategories remained.

In both tasks equally, ADVP and PP distinctions were mostly eliminated. The same goes for VP distinctions although VPs with parent node S, SBAR and VP had a higher tendency to remain unmerged.

These results indicate that by far the most important parent node information for both NP identification and chunking are the parent nodes of the NPs themselves. More detailed analysis of merge sets would be needed to see what exactly this means.

[12]ADJP: 6, ADVP: 18, FRAG: 2, INTJ: 2, NAC: 4, NP: 23, NX: 2, PP: 28, S: 14, SBAR: 20, SBARQ: 3, SINV: 2, UCP: 8, VP: 3.

4. Conclusions and Further Research

Grammar Learning by Partition Search was shown to be an efficient method for constructing PCFGs optimised for a given parsing task. In the nonlexicalised applications reported in this paper, the performance of the base grammar was improved by up to 3.48%. This corresponds to an improvement of up to 21.45% over the standard baseline. The result for NP chunking is slightly better than the best existing result for nonlexicalised NP chunking, whereas the result for NP identification closely matches the best existing result for lexicalised NP identification.

Partition Search can also be used to simply reduce grammar size, if an existing grammar is used as the Max Grammar. In the experiments reported in this paper, Partition Search reduced the size of nonterminal sets by up to 93.5%, and the size of rule sets by up to 27.4%. Compared to other grammar compression techniques, it has the advantage of being lossless.

Further research will look at additionally incorporating lexicalisation, other search methods, and other variable parameter combinations.

5. Acknowledgements

The research reported in this paper was in part funded under the European Union's TMR programme (Grant No. ERBFMRXCT980237).

6. References

Steven Abney. 1991. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle-Based Parsing*, pages 257–278. Kluwer Academic Publishers, Boston.

A. Belz. 2001. Optimising corpus-derived probabilistic grammars. In *Proceedings of Corpus Linguistics 2001*, pages 46–57.

A. Belz. 2002. Grammar learning by partition search. In *Proceedings of LREC Workshop on Event Modelling for Multilingual Document Linking*.

Eugene Charniak and Glenn Carroll. 1994. Context-sensitive statistics for improved grammatical language models. Technical Report CS-94-07, Department of Computer Science, Brown University.

Eugene Charniak. 1996. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

A. J. Korenjak. 1969. A practical method for constructing LR(*k*) processors. *Communications of the ACM*, 12(11).

A. Krotov, M. Hepple, R. Gaizauskas, and Y. Wilks. 2000. Evaluating two methods for treebank grammar compaction. *Natural Language Engineering*, 5(4):377–394.

J. Nerbonne, A. Belz, N. Cancedda, Hervé Déjean, J. Hammerton, R. Koeling, S. Konstantopoulos, M. Osborne, F. Thollard, and E. Tjong Kim Sang. 2001. Learning computational grammars. In *Proceedings of CoNLL 2001*, pages 97–104.

H. Schmid and S. Schulte Im Walde. 2000. Robust German noun chunking with a probabilistic context-free grammar. In *Proceedings of COLING 2000*, pages 726–732.

H. Schmid. 2000. LoPar: Design and implementation. Bericht des Sonderforschungsbereiches "Sprachtheoretische Grundlagen für die Computerlinguistik" 149, Institute for Computational Linguistics, University of Stuttgart.

E. Tjong Kim Sang, W. Daelemans, H. Déjean, R. Koeling, Y. Krymolowski, V. Punyakanok, and D. Roth. 2000. Applying system combination to base noun phrase identification. In *Proceedings of COLING 2000*, pages 857–863.

Jose Luis Verdú-Mas, Jorge Calera-Rubio, and Rafael C. Carrasco. 2000. A comparison of PCFG models. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 123–125.

F. L. Weng and A. Stolcke. 1995. Partitioning grammars and composing parsers. In *Proceedings of the 4th International Workshop on Parsing Technologies*.

An integration of Vector-Based Semantic Analysis and Simple Recurrent Networks for the automatic acquisition of lexical representations from unlabeled corpora

Fermín Moscoso del Prado Martín[*], **Magnus Sahlgren**[†]

[*]Interfaculty Research Unit for Language and Speech (IWTS)
University of Nijmegen & Max Planck Institute for Psycholinguistics
P.O. Box 310, NL-6500 AH Nijmegen, The Netherlands
fermin.moscoso-del-prado@mpi.nl

[†]Swedish Institute for Computer Science (SICS)
Box 1263, SE-164 29 Kista, Sweden
mange@sics.se

Abstract

This study presents an integration of Simple Recurrent Networks to extract grammatical knowledge and Vector-Based Semantic Analysis to acquire semantic information from large corpora. Starting from a large, untagged sample of English text, we use Simple Recurrent Networks to extract morpho-syntactic vectors in an unsupervised way. These vectors are then used in place of random vectors to perform Vector-Space Semantic Analysis. In this way, we obtain rich lexical representations in the form of high-dimensional vectors that integrate morpho-syntactic and semantic information about words. Apart from incorporating data from the different levels, we argue how these vectors can be used to account for the particularities of each different word token of a given word type. The amount of lexical knowledge acquired by the technique is evaluated both by statistical analyses comparing the information contained in the vectors with existing 'hand-crafted' lexical resources such as CELEX and WordNet, and by performance in language proficiency tests. We conclude by outlining the cognitive implications of this model and its potential use in the bootstrapping of lexical resources

1. Introduction

Collecting word-use statistics from large text corpora has proven to be a viable method for automatically acquiring knowledge about the structural properties of language. The perhaps most well-known example is the work of George Zipf, who, in his famous *Zipf's laws* (Zipf, 1949), demonstrated that there exist fundamental statistical regularities in language. Although the useability of statistics for extracting structural information has been widely recognized, there has been, and still is, much scepticism regarding the possibility of extracting semantic information from word-use statistics. We believe that part of the reason for this scepticism is the conception of meaning as something external to language — as something *out there* in the world, or as something *in here* in the mind of a language user. However, if we instead adopt what we may call a "Wittgensteinian" perspective, in which we do not demand any rigid definitions of word meanings, but rather characterize them in terms of their use and their "family resemblance" (Wittgenstein, 1953), we may argue that word-use statistics provide us with exactly the right kind of data to facilitate semantic knowledge acquisition. The idea, first explicitly stated in Harris (1968), is that the meaning of a word is related to its distributional pattern in language. This means that if two words frequently occur in similar context, we may assume that they have similar meanings. This assumption is known as "the Distributional Hypothesis," and it is the ultimate rationale for statistical approaches to semantic knowledge acquisition, such as Simple Recurrent Networks or Vector-Based Semantic Analysis.

1.1. Simple Recurrent Networks

Simple Recurrent Networks (SRN; Elman, 1990) are a class of Artificial Neural Networks consisting of the three traditional 'input', 'hidden' and 'output' layers of units, to which one additional layer of 'context' units is added. The basic architecture of an SRN is shown in Figure 1. The outputs of the 'context' units are connected to the inputs of the 'hidden' layer as if they formed and additional 'input' layer. However instead of receiving their activation from outside, the activations of the 'context' layer at time step n are a copy of the activations of the 'hidden' layer at time step $n - 1$. This is achieved by adding simple, one-to-one 'copy-back' connections from the 'hidden' layer into the 'context' layer. In contrast to all the other connections in the network, these are special in that they are not trained (their weights are fixed at 1), and in that they perform a raw copy operation from a hidden unit into a context unit, that is to say, they employ the identity function as the activation function. Networks of this kind combine the advantages of recurrent networks, their capability of maintaining a history of past events, with the simplicity of multilayer perceptrons as they can be trained by the backpropagation algorithm.

Elman (1993) trained an SRN on predicting the next word in a sequence of words, using sentences generated by an artificial grammar, with a very limited vocabulary (24 words). He showed that a network of this class, when trained on a word prediction task and given the right training strategy (see (Rohde and Plaut, 2001) for further discussion of this issue), acquired various grammatical properties such as verbal inflection , plural inflection of nouns, argumental structure of verbs or grammatical category. More-

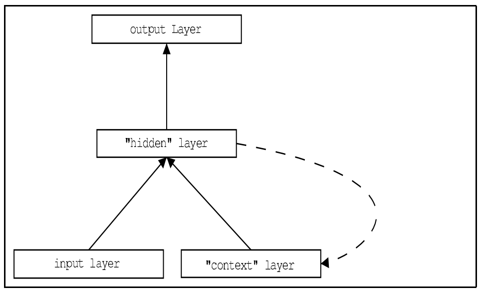


Figure 1: Modular architecture of a Simple Recurrent Network. The boxes correspond to layers of units. The solid arrows represent sets of trainable 'all-to-all' connections between the units in two layers. The dashed arrow stands a for fixed 'one-to-one' not trainable connection between two layers. These connections have the function of copying the activation of the hidden units into the context units at every time step.

over, the activations of the hidden units of the network provided detailed, token-specific characterizations of the morpho-syntactic properties of a word. Moscoso del Prado and Baayen (2001) showed how this method can be extended to deal with the large vocabulary sizes of realistic corpora. They trained a network akin to those of (Rohde and Plaut, 1999; Elman, 1993) on a word prediction task, using moderately large corpora of written English and Dutch (approximately 700,000 tokens of English and 4,500,000 of Dutch). The hidden units again provided rich representations of the morpho-syntactic properties of the words, containing information ranging from grammatical category, to subtle inflectional details such as verbal inflection or adjective gender. Moreover, the network had also captured some semantic properties of words, namely semantic properties that can be inferred from syntactic properties such as argumental structure.

1.2. Vector-Based Semantic Analysis

While SRN's appear to be more sensitive towards syntactic features, vector-space models have been used for over a decade to acquire and represent semantic information about words, documents and other linguistic units. This is done by collecting co-occurrence information in a words-by-contexts matrix F where each row F_w represents a unique word and each column F_c represents a context, which can either be a multi-word segment such as a document or another word. Latent Semantic Analysis/Indexing (LSA/LSI; Deerwester et al., 1990; Landauer and Dumais, 1997) uses document-based co-occurrence statistics, while Hyperspace Analogue to Language (HAL; Lund et al., 1995) and Schutze (1992) use word-based statistics. The cells of the matrix indicate the (weighted and/or normalized) frequency of ocurrence in, or co-occurrence with, the co-occurrence context (i.e. documents or words). Vector-space models generally also use some form of dimension reduction to reduce the computational strains of dealing with the rather ungainly co-occurrence matrix. LSA uses Singular Value Decomposition (SVD) and HAL uses a "column variance method," which consists in discarding the columns with lowest variance. This reduces the dimensionality of the co-occurrence matrix to a fraction of its original size. linguistic units are thus represented in the final reduced matrix by semantic vectors of n dimensionality. LSA is reported to be optimal at $n = 300$ (Landauer and Dumais, 1997), HAL at $n = 200$ (Lund et al., 1995), while Schutze (1992) use $n = 20$. A different approach to create the vector-space is Random Indexing (Kanerva et al., 2000; Karlgren and Sahlgren, 2001), which avoids the inefficient and inflexible dimensionality reduction phase by using high-dimensional sparse, random *index vectors* to accumulate a words-by-contexts matrix in which words are represented by high-dimensional (i.e. n is in the order of thousands) *context vectors*.

Vector-space methodology has been empirically validated in a number of experiments as a viable technique for the automatic extraction of semantic information from raw, unstructured text data. For example, Landauer and Dumais (1997) report a result on a standardized vocabulary test (TOEFL; Test of English as a Foreign Language) that is comparable to the average performance of foreign (non-English speaking) applicants to U.S. colleges (64.4% vs. 64.5% correct answers to the TOEFL). Sahlgren (2001), showed that similar performance (64.5% – 67%) may be obtained by using distributed representations in the Random Indexing technique that eliminates the need for the computationally expensive SVD, and he also demonstrated that the performance may be further enhanced (72% correct answers) by taking advantage of explicit linguistic information (morphology). Further empirical evidence can be found in, for example, (Lund and Burgess, 1996), who used semantic vectors to model reaction times from lexical priming studies, and from (Landauer and Dumais, 1997), who used LSA for evaluating the quality of content of student essays on given topics. Thus, it appears to be beyond doubt that the vector-space methodology really is able to form high-quality semantic representations by using such a simple souce of information as plain co-occurrence statistics. In the remainder of this paper, we will use the label *Vector-Based Semantic Analysis* to denote the practice of using co-occurrence information to construct vectors representing linguistic units in a high-dimensional semantic space.

2. Goal of the paper

In this study, we integrate two techniques to automatically obtain distributed lexical representations from corpora encoding morpho-syntactic and semantic information simultaneously. A hybrid technique such as the one that we describe here has several advantages. First, it requires a minimum of preexisting lexical resources, as it depends only on raw corpora. There is no need for taggers or parsers which, for many languages, may be unavailable.

Second, in contrast to other approaches that exploit word co-occurrences, our method keeps computational costs under control, as we avoid having to deal with huge co-occurrence matrices and we do not need to apply dimensional reduction techniques such as Singular Value Decomposition or Principal Component Analysis. The use of such dimensional reduction techniques imposes important limitations on the extension of existing resources, as the addition of a new item would requires that a new reduced similarity space is calculated. In contrast, both SRN and the VBSA technique allow for the direct inclusion of new data. Another important advantage of our approach is that lexical representations become dynamic in nature: each token of a given type will have a slightly different representation.

We produce explicit measures of reliability that are directly associated to each distance calculated by our method. This is particularly useful for extending existing lexical resources such as computational thesauri.

In what follows, we introduce the corpus employed in the experiment, together with the SRN and VBSA techniques that we used. We then evaluate the grammatical knowledge encoded in the distributed representations obtained by the model. We subsequently evaluate the semantic knowledge contained in the system by means of scores on language proficiency tests (TOEFL), comparison with synonyms in WordNet, and a comparison of the properties of morphological variants. We conclude by discussing the possible application of this technique to bootstrap lexical resources from untagged corpora and the cognitive implications of these results.

3. The Experiment

3.1. Corpus

For the training of the SRN network, we used the texts corresponding to the first 20% of the British National Corpus; by first we mean that we selected the files following the order of directories, and we included the first two directories in the corpus. This corresponds to roughly 20 million tokens. To allow for comparison with the results from (Sahlgren, 2001), which were based on a 10 million word corpus, only the first half of this subset was used in the application of the VSBA technique.

Only a naive preprocessing stage was performed on the original SGML files. This included removing all SGML labels from the corpus, converting all words to lower case, substituting all numerical tokens for a $[num]$ token and separating hyphenated compound words into three different tokens ($first\ word + [hyphen] + second\ word$). All tokens containing non alphabetic characters different from the common punctuation marks were removed from the corpus. Finally, to reduce the vocabulary size, all tokens that were below a frequency threshold of two, were substituted by an $[unknown]$ token.

3.2. Design and training of the SRN

The Simple Recurrent Network followed the basic design shown in Figure 1. We used a network with 300 units in the input and output layers, and 150 units in the hidden and context layers. To allow for representation of a very large number of tokens, we used the semi-localist approach described in (Moscoso del Prado and Baayen, 2001) with a code of three random active units per word. On the one hand, this approach is close to a traditional style one-bit-per-word localistic representation in that the vectors of two different words will be nearly orthogonal. The small deviation from full orthogonality between representations has an effect similar to the introduction of a small amount of random noise, which actually speeds up the learning process. On other the hand, using semi-distributed input/output representations allows us to represent a huge number of types (a maximum of $\binom{300}{3} = 4,455,100$ types), while keeping the size of the network moderately small.

The sentences of the corpus were grouped into 'examples' of five consecutive sentences. At each time step, a word was presented to the input layer and the network would be trained to predict the following word in the output units. The corpus sentences were presented word by word in the order in which they appear. After every five sentences (a full 'example'), the activation of the context units was reset to 0.5. Imposing limitations on the network's memory on the initial stages of training is a pre-requisite for the networks to learn long distance syntactic relations (Elman, 1993; cf., Rohde and Plaut, 2001; Rohde and Plaut, 1999). We implemented this 'starting small' strategy by introducing a small amount of random noise (0.15) in the output of the hidden units, and by gradually reducing to zero during training. At the same time that the random noise in the context units was being reduced, we also gradually reduced the learning rate, starting with a learning rate of 0.1 and finished training with a learning rate of 0.4. Throughout training, we used a momentum of 0.9.

Although the experiments in (Elman, 1993) used the traditional backpropagation algorithm, using the mean square error as the error measure to minimize, following (Rohde and Plaut, 1999) we substituted the training algorithm for a modified momentum descent using cross-entropy as our error measure,

$$\sum_i \left[t_i \log\left(\frac{t_i}{o_i}\right) + (1 - t_i) \log\left(\frac{1 - t_i}{1 - o_i}\right)\right] \qquad (1)$$

Modified momentum descent enables stable learning with very aggressive learning rates as the ones we use. The network was trained on the whole corpus of 20 million for one epoch using the *Light Efficient Network Simulator* (LENS; Rohde, 1999).

3.3. Application of VBSA technique

Once the SRN had been trained, we proceeded to apply the Vector Based Semantic Analysis technique. Sahlgren (2001) used what he called 'random labels'. These were sparse 1800 element vectors, in which, for a given word type, only a small set of randomly chosen elements would be active (± 1.0), while the rest would be inactive. Once these initial labels had been created, the corpus was processed in the following way. For each token in the corpus, the labels of the s immediately preceding or following tokens were added to the vector of the word (all vectors were initialized to a set of 0's). The addition would be weighted giving more importance to the closer word in the window. Words outside a frequency range of $(3 - 14,000)$ are not included in these sums. This range excludes both the very frequent types, typically function words, and the least frequent types, about which there is not enough information to provide reliable counts. Optimal results are obtained with a window size ($s = 3$), that is, by taking into account the three preceeding and following words to a given token.

In order to reduce sparsity, Sahlgren used a lemmatizer to unify tokens representing inflectional variants of the same root. Sahlgren had also observed that the inclusion of explicit syntactic information extracted by a parser did not improve the results, but led to lower performance. We believe that this can be partly due to the *static* character of the syntactic information that was used. We therefore use a *dynamic* coding of syntactic information, which is more sensitive to the subtle changes in grammatical properties of each different instance of a word.

In our study, we substituted the knowledge-free random labels of (Sahlgren, 2001) by the dynamic context-sensitive representations of the individual tokens as coded in the patterns of activations of our SRN. Thus each type is represented by a slightly different vector for each different grammatical context in which it appears. To obtain these representation, we presented the text to the SRN and used the activation of the hidden units to provide the dynamic labels for VBSA

We then used a symmetric window of three words to the left and right of every word. We fed the text again through the neural network in test mode (no weight updating), and we summed the activation of the hidden units of the network for each of the words in the context window that fall within a frequency range of 8 and $30,000$ in the original corpus (the one that was used for the training of the neural network). In this way we excluded low frequency words about which the network might be extremely uncertain, and extremely high frequency function words. We used as weighting schema $w = 2^{1-d}$, were w is the weight for a certain position in the window, and d is the distance in tokens from that position to the center of the window. For instance, the label of the word following the target would be added with a weight $w = 2^{1-1} = 1$ and the label of the word occupying the leftmost position in the window would have a weight $w = 2^{1-3} = 0.25$. When a word in the window was out of the frequency range, its weight was set to 0.0. Punctuation marks were not included in window positions.

4. Results

4.1. Overview of semantics by nearest neighbors

We begin our analysis by inspecting the five nearest neighbors for a given word. Some examples can be found in Table 4.1. To calculate the distances between words, we use normalized cosines (Schone and Jurafsky, 2001). Traditionally, high dimensional lexical vectors have been compared using metrics such as the cosine of the angle between the vectors or the classical Euclidean distance metric or the city-block distance metric. However, using a fixed metric on the components of the vectors induces undesirable effects pertaining to the centrality of representations. More frequent words tend to appear in a much wider range of contexts. When the vectors are calculated as an average of all the tokens of a given type, the vectors or more frequent words will tend to occupy more central positions in the representational space. They will tend to be nearer to all other words, thus introducing an amount of relativity in the distance values. In fact, we believe that this relativity actually reflects people's understanding of word meaning. For example, if we considered the most similar words to a frequent word such as "bird", we would find words as "pigeon" to be very related in meaning. A word such as "penguin" would be considered a more distantly related word. However, if we examined the nearest neighbors of "penguin", we would probably find "bird" among them, although the standard distance measure would still be high. A way to overcome this problem is to place word distances inside a normal distribution, taking into account the distribution of distances of both words. Consider the classical cosine distance between two vectors $\mathbf{v}$ and $\mathbf{w}$:

$$d_{\cos}(\mathbf{v},\mathbf{w}) = 1 - \frac{\mathbf{v}\cdot\mathbf{w}}{||\mathbf{v}||\,||\mathbf{w}||}. \qquad (2)$$

For each vector $\mathbf{x} \in \{\mathbf{v},\mathbf{w}\}$ we calculate the mean ($\mu_{\mathbf{x}}$) and standard deviation ($\sigma_{\mathbf{x}}$) of its cosine distance to 500 randomly chosen vectors of other words. This provides us with an estimate of the mean and standard deviation of the distances between $\mathbf{x}$ and all other words. We can now define the normalized cosine distance between two vectors $\mathbf{v}$ and $\mathbf{w}$ as:

$$d_{norm}(\mathbf{v},\mathbf{w}) = \max_{\mathbf{x}\in\{\mathbf{v},\mathbf{w}\}} \left(\frac{d_{\cos}(\mathbf{v},\mathbf{w}) - \mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}\right). \qquad (3)$$

To speed up this process, the cosine distance means and standard deviation for all words were pre-calculated in advance and stored as part of the representation. The use of normalized cosine distance has the effect of allowing for direct comparisons of the distances between words. In our previous example the distance between "bird" and "penguin", according to a non-normalized metric would suffer from the eccentricity of "penguin"; with the normalization, as the value of the distance would be normalized with respect to "penguin" (the maximum), it would render a value similar to the distance between "bird" and "pigeon".

4.2. Grammatical knowledge

Moscoso del Prado and Baayen (2001) showed that the hidden unit representations of SRN's similar to the one we used here contain information about morpho-syntactic characteristics of the words. In the present technique this information is implicitly available in the input labels for the VBSA technique. The VBSA component however, does not guarantee the preservation of such syntactic information. We therefore need to ascertain whether the grammatical knowledge contained in the SRN vectors is preserved after the application of VBSA.

Note that in Table 4.1., the nearest neighbors of a given word tend to have similar grammatical attributes. For example, plural nouns have other plural nouns as nearest neighbors, e.g., "foreigners" - "others", "outsiders", etc., and verbs tend to have other verbs as nearest neighbors, e.g., "render" - "expose", "reveal", etc. Although the nearest neighbors in Table 4.1. clearly suggest that morpho-syntactic information is coded in the representations, we need to ascertain how much morpho-syntactic information is present and, more importantly, how easily it might be made more explicit. We do this using the techniques proposed in (Moscoso del Prado and Baayen, 2001), that is we employ a machine learning technique using our vectors

Word	Nearest neighbors
hall	centre, theatre, chapel, landscape*, library
half	period, quarter, phase, basis, breeze*
foreigners	others, people, doctors, outsiders, unnecessary*
legislation	orders, contracts, plans, losses, governments
positive	splendid, vital, poetic, similar*, bad
slightly	somewhat, distinctly, little, fake*, supposedly
subjects	issues, films, tasks, substances, materials
taxes	debts, rents, imports, investors, money
render	expose, reveal, extend, ignoring*, develop
re-	anti-, non-, pro-, ex-, pseudo-
omitted	ignored, despised, irrelevant, exploited*, theirs*
Bach	Newton, Webb, Fleming, Emma, Dante

Table 1: Sample of 5 nearest neighbors to some words according to normalized cosine distance. Semantically unrelated words are marked by an asterisk

as input and symbolic grammatical information extracted from the CELEX database (Baayen et al., 1995) as output. A machine learning system is trained to predict the labels from the vectors. The rationale behind this method is very straightforward: If there is a distributed coding of the morpho-syntactic features hidden inside our representation, a standard machine learning technique should be able to detect it.

We begin by assessing whether the grammatical category of a word can be extracted from its vector representation. We randomly selected 500 words that were classified by CELEX as being unambiguously nouns or verbs, that is, they did not have any other possible label. The nouns were sampled evenly between singular an plural nouns, and the verbs were sampled evenly between infinitive, third person singular and gerund forms. Using TiMBL (Daelemans et al., 2000), we trained a memory based learning system on predicting whether a vector corresponded to a noun or a verb. We performed ten-fold cross-validation on the 500 vectors. The systems were trained using 7 nearest neighbors according to a city-block distance metric, the contribution of each component of the vectors weighted by Information Gain Feature Weighting (Quinlan, 1993). To provide a baseline against which to compare the results, we use a second set of files consisting of the same vectors but with random assignment of grammatical category labels to words. The average performance of the system of the Noun-Verb distinction was 68% (randomized averaged 56%). We compared the performance of the system with that of the randomized labels system using a paired two-tailed t-test on the result of each of the runs in the cross-validation, which revealed that the performance of the system was significantly higher than that of the randomized one ($t = 5.63$, $df = 9$, $p = 0.0003$).

We also tested for more subtle inflectional distinctions. We randomly selected 300 words that were unambiguously nouns according to CELEX, sampling evenly from singular and plural nouns. We repeated the test described in the previous paragraph, with the classification task this time being the differentiation between singular and plural. The average performance of the machine learning system was 65% (randomized averaged 48%). A paired two-tailed t-test comparing the results of the systems with the results of systems with the labels randomized revealed again a significant advantage for the non-random system ($t = 5.80$, $df = 9, p = 0.0003$). The same test was performed on a group of 300 randomly chosen unambiguous verbs sampled evenly among infinitive, gerund and third person singular forms, with these labels being the ones the system should learn to predict from the vectors. Performance in differentiating these verbal inflections was of 55% on average while the average of randomized runs was 33%, and significantly above randomized performance accoriding to a paired two-tailed t-test ($t = 4.25$, $df = 9$, $p = 0.0021$).

4.3. Performance in TOEFL synonyms test

Previous studies (Sahlgren, 2001; Landauer and Dumais, 1997) evaluated knowledge about semantic similarity contained in co-occurrence vectors by assessing their performance in a vocabulary test from the Test of English as a Foreign Language (TOEFL). This is a standardized vocabulary test employed by, for instance, American universities, to assess foreign applicants' knowledge of English. In the synonym finding part of the test, participants are asked to select which word is a synonym of another given word, given a choice of four candidates that are generally very related in meaning to the target. In the present experiment, we used the selection of 80 test items described in (Sahlgren, 2001), with the removal of seven test items which contained at least one word that was not present in our representation. This left us with 73 test items consisting of a target word and four possible synonyms. To perform the test, for each test item, we calculated the normalized cosine distance between the target word and each of the candidates, and chose as a synonym the candidate word that showed the smallest cosine distance to the target. The model's performance on the test was 51% of correct responses.

4.3.1. Reliability scores

The results of this test can be improved once we have a measure of the certainty with which the system considers the chosen answer to be a synonym of the target. What we need is a reliability score, according to which, in cases where the chosen word is not close enough in meaning, i.e., its distance to the target is below a certain probabilistic threshold, the system would refrain from answering. In other words, the system would be allowed to give an answer such as: "I'm not sure about this one". Given that the values of the distances between words in our system, follow a normal distribution $\mathcal{N}(0,1)$, it is quite straightforward to obtain an estimate of the probability of the distance between two words being smaller than a given value, by just using the Normal distribution function $F(x)$. However, while the *general* distribution of distances between any two given words follows $\mathcal{N}(0,1)$, the distribution of the distances from a *particular* word to the other words does not necessarily follow this distribution. In fact they generally do not do so. This difference in the distributions of distances of words is due to effects of prototypicality and probably also word frequency (McDonald and Shillcock, 2001).

To obtain probability scores on how likely it is that a given word is at a certain distance from the target, we need to see the distance of this word *relative* to the distribution of distances from the target word to all other words in the representation. We therefore slightly modify 3, which takes the normalized distance between two words to be the maximum of the cosine distance normalized according to the distribution of distances to the first word, and the cosine distance normalized to the distribution of distances to the second word. We now define the cosine distance between two vectors $\mathbf{v}$ and $\mathbf{w}$ normalized relative to $\mathbf{v}$ as:

$$d^{\mathbf{v}}_{norm} = \frac{d_{\cos}(\mathbf{v},\mathbf{w}) - \mu_{\mathbf{v}}}{\sigma_{\mathbf{v}}}, \quad (4)$$

which provides us with distances that follow $\mathcal{N}(0,1)$ for each particular word represented by a vector $\mathbf{v}$.

Using 4, we calculated the distance between the target words in the synonym test and the word that the system had selected as most similar, counting only those answers for which the system outputs a probability value below 0.18. The performance on the test increases from 51% to 71%, but the number of items reduced to 45. If we choose probability values below 0.18, the percentage correct continues to rise, but the number of items in the test drops dramatically. Having such a reliability estimator is useful for real-world applications.

4.4. Performance for WordNet synonyms

We can also use the WordNet (Miller, 1990) lexical database to further assess the amount of word similarity knowledge contained in our representations. We randomly selected synonym pairs from each of the four grammatical categories contained in WordNet: nouns, verbs, adjectives and adverbs. We calculated the normalized cosine distance for each of the synonym pairs. As expected, the median distances between synonymous words were clearly smaller than average distance. The median distances were -0.59 for verb synonyms, -0.53 for noun synonyms, -0.49 for adjective synonyms and -0.62 for adverbial synonyms. However, as we have already seen, our vectors contain a great deal of information about morpho-syntactic properties. Hence the fact that synonyms share the same grammatical category could by itself explain the small distances obtained for WordNet synonyms. To check whether this is the case, each synonym pair from our set was coupled with a randomly chosen baseline word of the same grammatical category, and we calculated the distance between one of the synonyms and the baseline word. In this case, as we were interested in the distance of the word relative only to one of the words in the pair, we calculated distances using 4. We compared the series of distances obtained for the true WordNet synonym pairs with the baseline distances by means of two-tailed t-tests. We found that WordNet synonyms were clearly closer in all the cases: nouns ($t = -5.30$, $df = 197$, $p < 0.0001$), verbs ($t = -4.60$, $df = 190$, $p < 0.0001$), adjectives ($t = -3.09$, $df = 195$, $p = 0.0023$) and adverbs ($t = -4.06$, $df = 188$, $p < 0.0001$). This shows that true synonyms were significantly closer in distance space than baseline words.

4.5. Morphology as a measure of meaning

Morphologically related words tend to be related both in form and meaning. This is true both for inflectionally related words, and derivationally related words. As morphological relations tend to reflect regular correspondences to slight changes in the meaning and syntax, they can be used for assessing the amount of semantic knowledge that has been acquired by our system. In what follows, we investigate whether our system is able to recognize inflectional variants of the same word, and whether the vectors of words belonging to the same suffixation class cluster together.

4.5.1. Inflectional morphology

We randomly selected 500 roots that were unambiguously nominal (they did not appear in the CELEX database under any other grammatical category) and for which both the singular and the plural form were present in our dataset. For each of the roots, we calculated the normalized cosine distance between the singular and plural forms. The median of the distance between singular and plural forms was -0.39, which already indicates that inflectional variants of the same noun are represented by similar vectors. As in the case of the WordNet synonyms, it could be argued that this below average distance is completely due to all these word pairs sharing the "noun" property. To ascertain that the observed effect on the distances was at least partly due to real similarities in meaning, each stem r_1 in our set was paired with another stem r_2 also chosen from the original set of 500 nouns. We calculated the normalized cosine distance between the singular form of r_1 and the plural form of r_2. In this way we constructed a data set composed of word pairs plus their normalized cosine distance. A linear mixed effect model (Pinheiro and Bates, 2000) fit to the noun data with normalized cosine distance as dependent variable, the 'stem' (same v. other) as independent variable and the root of the present tense form as random effect, revealed a main effect for stem-sharing pairs ($F(1,499) = 44.42$, $p < 0.0001$). The coefficient of the effect was -0.29 ($\hat{\sigma} = 0.043$). This indicates that the distances between pairs of nouns that share the same stem are in general smaller than the distance between pairs of words that do not share the same root but have the same number. Interestingly, according to a Pearson correlation, 65% of the variance in the distances is explained by the model.

In the same way, we randomly selected 500 unambiguously verbal roots for which we had the present tense, past tense, gerund and third person singular present tense in our representation. The median normalized cosine distance between the present tense and the other forms of the verb was -0.48, so verbs seem to be clustered together somewhat more tightly than nouns. We repeated the test described above by random pairing of stems, but now we calculated the distances between the present tense form of r_1 and the rest of the inflected forms of r_2. We fit a linear mixed effect model with the normalized cosine distance between the pairs as dependent variable, the pair of inflected forms, i.e., *present-past*, *present-gerund*, or *present-third person singular*, and the 'stem' (same versus different) as independent variables and the root of the first verb as random effect. We found significant, independent effects for type of inflectional pair ($F(1, 2495) = 289.06$, $p < 0.0001$) and stem-sharing ($F(1, 2495) = 109.76$, $p < 0.0001$). The interaction between both independent variables was not significant ($F < 1$). The coefficient for the effect of sharing a root was -0.18 ($\hat{\sigma} = 0.017$), which again indicates that words that share a root have smaller distances than words that do not. It is also interesting to observe that the coefficients for the pairs of inflected forms also provide us with information of how similarly these forms are used in natural language, or, phrased in another way, how similar their meanings are. So, the value of the coefficient for pairs of present tense (uninflected) and past tense forms was -0.48 ($\hat{\sigma} = 0.21$) and the coefficient for pairs composed of a present tense uninflected form and a past tense was -0.38 ($\hat{\sigma} = 0.21$), which suggests that the contexts in which an un-inflected form is used are more similar to the contexts where a past tense form is used than to the contexts of a gerund. The model explained 43% of the variance according to a Pearson correlation.

4.5.2. Derivational morphology

Derivational morphology also captures regular meaning changes, although these changes are often not as regular as the ones that are carried out by inflectional morphology. We tested whether our system captures derivational semantics using the Memory-Based Learning technique that we used for evaluating grammatical knowledge in the system (see section 4.2.). Concentrating on morphological categories, i.e. on words that share the same outer affix. For instance "compositionality" belongs to the morphological category "-ity" and not to the category "-al", although it also contains the suffix "-al". Derivational suffixes generally effect both syntactic and semantic changes. To test whether our vectors reflect semantic regularities, we selected all words ending in the two derivational suffixes "-ist" and "-ness". Both of these suffixes produce nouns, but while the first one generates nouns that are considered agents of actions, the second generates abstract ideas. These affixes generate words with the same grammatical category, but with different semantics. We trained a TiMBL system on predicting the morphological category of the vectors, that is, to predict "-ist" or "-ness". The average performance of the system in predicting these labels in a ten-fold cross-validation was of 78% (compared to an average of 51% obtained when randomizing the affix labels). A paired two-sided t-test between the system performance at each run and the performance of a randomized system on the same run, revealed a significant improvement for the non random system ($t = 10.95$, $df = 9$, $p < 0.0001$).

Although performance was very good for these two nominal affixes, a similar comparison between the adjectival affixes "-able" and "-less", did not render significant differences between randomized and non-randomized labels, indicating that the memory-based learning system was not able to discriminate these two affixes on the sole basis of their semantic vectors. This indicates that, although some of the semantic variance produced by derivational affixes can be captured, many subtler details are being overlooked.

5. Discussion

The analyses that we have performed on the vectors indicate that a high amount of lexical information has been captured by the combination of an SRN with VBSA. On the one hand, the results reported in section 4.2. indicate that the morpho-syntactic information that is coded in the hidden units of a SRN is maintained after the application of VSBA. Moreover, it is clear that the coding of the morpho-syntactic features can be extracted using a standard machine-learning technique such as Memory-Based Learning. This, by itself can be of great use in the bootstrapping of language resources. Given a fairly small set of words that have received morpho-syntactic tags, it is possible to train a machine learning system to identify these labels from their vectors, and then apply this to the vectors of words that are yet to receive morpho-syntactic tagging. Importantly, our technique relies only on word-external order and co-occurrence information, but does not make use of word-internal form information. As it it is evident that word-form information such as presence of inflectional affixes is crucial for morpho-syntactic tagging, our technique can be used to provide a confirmation of possible inflectional candidates. For instance, suppose that two words such as "rabi" and "rabies" are found in a corpora, one would be inclined to classify them as singular and plural version of the same word, when in fact they are both singular forms. The inflectional information in our vectors could be used to disconfirm this hypothesis. In this same aspect, the fact that inflectional variants of the same root tend to be very related in meaning could be used as additional evidence to reject this pair as being inflectional variants.

On the other hand, the nearest neighbors, the TOEFL scores, the results on detecting inflectionally and derivationally related words, and the results on the WordNet synonyms, provide solid evidence that the vectors have succeeded in capturing a great deal of semantic information. Although it is clear to us that our technique needs further fine-tuning, the results are already surprising given the constraints that have been imposed on the system. For instance, the performance on the TOEFL test (51% without the use of the Z scores) is certainly lower than many results that have been reported in the literature. Sahlgren (2001), using the Random Indexing approach to VBSA with random vectors reports 72% correct responses on the same test items. However, he was using a tagged corpus where all inflectional variants had been unified under the same type. Without the use of stemming, the best performance he reports is of 68%. In the current approach we have used vectors of 150 elements, that is, less than 10% of the size of the vectors used by Sahlgren, and much smaller than the vectors needed to apply techniques such Hyperspace Analog to Language (Lund et al., 1995; Lund and Burgess, 1996) or Latent Semantic Analysis (Landauer and Dumais, 1997) which need to deal with huge co-occurrence matrices. Given the computational requirements of using such huge vectors, we consider that our method provides a good alternative. Our result of 51% on the TOEFL test is clearly above chance performance (25%) and not that far from the results obtained by average foreign applicants to U.S. universities (64.5%). Interestingly, Landauer and Dumais (1997) reported a 64.4% performance on these test items using LSA, but this was only after the application of a dimensional reduction technique (SVD) to their original document co-occurrence vectors. Before the application of SVD, they report a performance of 36% on the plain normalized vectors. Of course, a technique such as SVD could be subsequently applied to the vectors obtained by our method, probably leading to some improvement in our results. However, given that our vectors already have a moderate size, and especially, given that, in their current state, one does not need to re-compute them to add information contained in new corpora, we do not favor the use of such techniques.

Regarding the evaluation of the system against synonym pairs extracted from the WordNet database, although the vectors represent synonyms as being more related than average, it still seems that most of the similarity in these cases was due to morpho-syntactic properties (the average difference in distances between the synonym and baseline conditions was always smaller than 0.1). We believe this is due to several factors. WordNet synonym sets (*synsets*) contain an extremely rich amount of information, that may be too rich for the purposes of evaluating our current vectors. First, many WordNet synonyms correspond to plain spelling variants of the same word in British and American English, e.g., "analyze"-"analise". Our whole training corpus was composed of British English, so the representation of words in American spelling is probably not very accurate. Second, and more importantly, given that the synsets encoded in WordNet reflect in many cases rare or even metaphoric uses of words, we think that the evaluation based on the average type representations provided by our system are not the most appropriate to detect these relations. Possibly, evaluating these synonyms against the vectors corresponding to the particular tokens referring to those senses might be more appropriate. An indication of this is also given by the TOEFL scores, which reflect that the meaning differences can still be detected in many cases. This is important because the synonyms pairs chosen in the TOEFL test, generally reflect the more standard senses of the words involved.

Another important issue is the difference between meaning *relatedness* and meaning *similarity*. These are two different concepts that appear to be somewhat confounded. While our representations reflect in many cases similarity relations, e.g, synonymy, they also appear to capture many relatedness and general world knowledge relations, for instance, the three nearest neighbors of "student" are "university" "pub" and "study", none of which is similar in meaning to "student", but all of them bearing a strong relationship to it. Sahlgren (2001) argues that using a small window to compute the co-occurrences (3 elements to each side, as compared to the 10 elements used in (Burgess and Lund, 1998)), has the effect of concentrating on similarity relations instead of relatedness, which would need much larger contexts such as the full documents used in LSA. The motivation to use very small context windows was to provide an estimation of the syntactic context of words. However, since syntactic information is already made more explicit by our SRN this may not be necessary in our case, and using larger window sizes might actually improve our performance both in similarity and relatedness. A further improvement that should be added to our vectors should come from the inclusion of word internal information. In a pilot experiment we have used the VBSA technique using (automatically constructed) distributed representations of the formal properties of words instead of the random labels. Performance on the TOEFL test were in the same range that was reported here (49%). This suggest that a combination of the technique described here with the formal vectors could probably provide much more precise semantic representations, exploiting both word internal and internal sources of information. This is also in line with the improvement of results found by (Sahlgren, 2001) when using a stemming technique. The use of formal vectors provides an interesting alternative, as it would supply implicit stemming information to the system.

In this paper, we have presented a representation that encodes jointly morpho-syntactic and semantic aspects of words. We have also provided evidence on how morphology is an important cue to meaning, and vice-versa, meaning is also an important cue to morphology. This corroborates previous results from (Schone and Jurafsky, 2001). The idea of integrating formal, syntactic and semantic knowledge about words in one single representation is currently gaining strength within the psycholinguistic community (Gaskell and Marslen-Wilson, 2001; Plaut and Booth, 2000). Some authors are considering morphology as the "convergence of codes", that is, as a set of quasi-regular correspondences between form and meaning, that would probably be linked at a joint representation level (Seidenberg and Gonnerman, 2000). Clear evidence of this strong link has also been put forward by (Ramscar, 2001) showing that the choice of regular or non-regular past tense inflection of a nonce verb is strongly influenced by the context in which the nonce verb appears. If the word appears in a context which entails a meaning similar to that of an irregular verb that is also similar in form to the nonce word, e.g. "frink" - "drink", participants form its past tense in the same manner as the irregular form, e.g., "frank" from "drank". If it appears in a context alike to a similar regular verb, e.g, "wink", participants inflect in regularly, e.g. "frinked" from "winked". Crucially, the meaning of this form is totally determined by context. This in line with the results of (McDonald and Ramscar, 2001), which show how the meaning of a nonce word is modulated by the context in which it appears. In this respect, our vectors constitute a first approach to such kind of representation: they include contextual and syntactic information. A further step will be the inclusion of word form information in this system, which is left for future research. Our lexical representations are formed by accumulation of predictions. On the one hand, several authors are currently investigating the strong role played by anticipation and prediction in human cognitive processing (e.g., Altmann, 2001). On the other hand, some current models of human lexical processing include the notion of accumulation, generally by recurrent loops in the semantic representations (e.g., Plaut and Booth, 2000).

Acknowledgments We are indebted to Harald Baayen and Rob Schreuder for helpful discussion of the ideas and techniques described in this paper.

The first author was supported by the Dutch Research Council (NWO) through a PIONIER grant awarded to R. Harald Baayen. The second author is funded through the DUMAS project, supported by the European Union IST Programme (contract IST-2000-29452).

6. References

Gerry Altmann. 2001. Grammar learning by adults, infants, and neural networks: A case study. In *7th Annual Conference on Architectures and Mechanisms for Language Processing AMLaP-2001*, Saarbrücken, Germany.

R. Harald Baayen, Richard Piepenbrock, and Léon Gulikers. 1995. *The CELEX lexical database (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.

C. Burgess and K. Lund. 1998. The dynamics of meaning in memory. In E. Dietrich and A. B. Markman, editors, *Cognitive dynamics: Conceptual change in humans and machines*. Lawrence Erlbaum Associates, Mahwah, NJ.

Walter Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2000. TiMBL: Tilburg Memory Based Learner Reference Guide. Version 3.0. Technical Report ILK 00-01, Computational Linguistics Tilburg University, March.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407.

J. L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14:179–211.

J. L. Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99.

M. Gareth Gaskell and William D. Marslen-Wilson. 2001. Representation and competition in the perception of spoken words. *(in press) Cognitive Psychology*.

Z. Harris. 1968. *Mathematical Structures of Language*. New York: Interscience publishers.

P. Kanerva, J. Kristofersson, and A. Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036. Mahwah, New Jersey: Erlbaum.

J. Karlgren and M. Sahlgren. 2001. From words to understanding. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of real-world intelligence*, pages 294–308. Stanford: CSLI Publications.

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

K. Lund and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behaviour Research Methods, Instruments, and Computers*, 28(2):203–208.

K. Lund, C. Burgess, and R. A. Atchley. 1995. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 660–665, Hillsdale, NJ. Erlbaum.

Scott McDonald and Michael Ramscar. 2001. Testing the distributional hypothesis: The influence of context judgements of semantic similarity. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*.

Scott A. McDonald and Richard C. Shillcock. 2001. Rethinking the word frequency effect: The neglected role of distributional information in lexical processing. *Language and Speech*, 44(3):295–323.

G. A. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–312.

Fermín Moscoso del Prado and R. Harald Baayen. 2001. Unsupervised extraction of high-dimensional lexical representations from corpora using simple recurrent networks. In Alessandro Lenci, Simonetta Montemagni, and Vito Pirrelli, editors, *The Acquisition and Representation of Word Meaning*. Kluwer Academic Publishers (*forthcoming*).

J. C. Pinheiro and D. M. Bates. 2000. *Mixed-effects models in S and S-PLUS*. Statistics and Computing. Springer, New York.

D. C. Plaut and J. R. Booth. 2000. Individual and developmental differences in semantic priming: Empirical and computational support for a single mechanism account of lexical processing. *Psychological Review*, 107:786–823.

J. R. Quinlan. 1993. *Programs for Machine Learning*. Morgan Kauffmann, San Mateo, CA.

Michael Ramscar. 2001. The role of meaning in inflection: Why past tense doesn't require a rule. *(in press) Cognitive Psychology*.

Douglas L. T. Rohde and David C. Plaut. 1999. Language acquisition in the absence of explicit negative evidence: how important is starting small? *Cognition*, 72(1):67–109.

Douglas L. T. Rohde and David C. Plaut. 2001. Less is less in language acquisition. In P. Quinlan, editor, *Connectionist Modelling of Cognitive Development*. (in press) Psychology Press, Hove, U.K.

Douglas L. T. Rohde. 1999. LENS: The light, efficient network simulator. Technical Report CMU-CS-99-164, Carnegie Mellon University, Pittsburg, PA.

Magnus Sahlgren. 2001. Vector-based semantic analysis: Representing word meanings based on random labels. In Alessandro Lenci, Simonetta Montemagni, and Vito Pirrelli, editors, *The Acquisition and Representation of Word Meaning*. Kluwer Academic Publishers (*Forthcoming*).

Patrick Schone and Daniel Jurafsky. 2001. Knowledge free induction of inflectional morphologies. In *Proceedings of the North American Chapter of the Association for Computational Linguistics NAACL-2001*.

Hinrich Schütze. 1992. Dimensions of meaning. In *Proceedings of Supercomputing '92*, pages 787–796.

Mark S. Seidenberg and Laura M. Gonnerman. 2000. Explaining derivational morphology as the convergence of codes. *Trends in the Cognitive Sciences*, 4(9):353–361.

Ludwig Wittgenstein. 1953. *Philosophical Investigations*. Oxford, Blackwell.

G. K. Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

Detection of Errors in PoS-Tagged Corpora by Bootstrapping Generalized Negative *n*-grams

Pavel Květoň and Karel Oliva
Austrian Research Institute for Artificial Intelligence (ÖFAI)
Schottengasse 3, A-1010 Wien, Austria
{pavel,karel}@oefai.at

Abstract

This paper presents two simple yet in practice very efficient bootstrapping techniques serving for automatic detection of those positions in a part-of-speech tagged corpus where an error is to be suspected. The first approach is based on the idea of stepwise learning and application of "negative bigrams", i.e. on the search for pairs of adjacent tags which constitute an incorrect configuration in a text of a particular language (in English, e.g., the bigram *ARTICLE - FINITE VERB*). As the second technique, the paper describes the stepwise generalization of the "negative bigrams" into "negative *n*-grams" (for increasing $n \in N$) which indeed provides a powerful tool for error detection in a corpus. The evaluation of results of the approach when used for error detection in the NEGRA corpus of German and the general implications for the quality of results of statistical taggers are also discussed. Illustrative examples in the text are taken from German, and hence at least a basic command of this language would be helpful for their understanding - due to the complexity of the necessary accompanying explanation, the examples are neither glossed nor translated. However, the central ideas of the paper should be understandable also without any knowledge of German.

1 Errors in PoS-Tagged Corpora

The importance of correctness (error-freeness) of language resources in general and of tagged corpora in particular cannot probably be overestimated. However, the definition of what constitutes an error in a tagged corpus depends on the intended usage of this corpus.

If we consider a quite typical case of a Part-of-Speech (PoS) tagged corpus used for training statistical taggers, then an error is defined naturally as any deviation from the regularities which the system is expected to learn; in this particular case this means that the corpus should contain neither errors in assignment PoS-tags nor ungrammatical constructions in the corpus body[1], since if any of the two cases is present in the corpus, then the learning process necessarily:

- gets a confused view of probability distribution of configurations (e.g., trigrams) in a correct text

and/or, even worse (and, alas, much more likely)

- gets positive evidence also about configurations (e.g., trigrams) which should not occur as the output of tagging linguistically correct texts, while simultaneously getting less evidence about correct configurations.

If we consider PoS-tagged corpora destinated for testing NLP systems, then obviously they should not contain any errors in tagging (since this would be detrimental to the validity of results of the testing) but on the other hand they should contain a certain amount of ungrammatical constructions, in order to test the behaviour of the tested system on a realistic input.

Both these cases share the quiet presupposition that the tagset used is linguistically adequate, i.e. it is sufficient for unequivocal and consistent assignment of tags to the source text[2].

As for using annotated corpora for linguistic research, it seems that even inadequacies in the tagset are tolerable provided they are marked off properly - in fact, these spots in the corpus might well be quite an important source of linguistic investigation since, more often than not, they constitute direct pointers to occurrences of linguistically "interesting" (or at least "difficult") constructions in the text.

2 Representativity

In corpus linguistics, the term representativity is understood as the representativity of a corpus wrt. kind of text or some phenomenon.

In this section, we intend to scrutinize the issue of representativity of a part-of-speech (PoS) tagged corpus wrt. to bigrams[3]. In this case, the phenomena[4] whose presence and relative frequency are at stake are:

- bigrams, i.e. pairs [First,Second] of tags of words occurring in the corpus adjacently and in this order
- unigrams, i.e. the individual tags.

We shall define the *qualitative representativity* wrt. bigrams as the kind of representativity meeting the following two complementary parts:

[1] In this paper we on purpose do not distinguish between "genuine" ungrammaticality, i.e. one which was present already in the source text, and ungrammaticality which came into being as a result of faulty conversion of the source into the corpus-internal format, e.g., incorrect tokenization, OCR-errors, etc.

[2] This problem might be – in a very simplified form – illustrated on an example of a tagset introducing tags for NOUNS and VERBS only, and then trying to tag the sentence *John walks slowly* - whichever tag is assigned to the word *slowly*, it is obviously an incorrect one. Natural as this requirement might seem, it is in fact not met fully satisfactorily in any tagset we know of; for more, cf. (Květoň and Oliva in prep.).

[3] The case of a trigrams, used more usual in tagging practice, would be almost identical but require more lengthy explanations. For the conciseness of argument, we limit the discussion to bigrams in most parts of the text.

[4] In an indeed broadly understood sense of the word "phenomenon".

- the representativity wrt. *the presence of all valid bigrams* of the language in the corpus, which means that if any bigram [First,Second] is a bigram in a correct sentence of the language, then such a bigram occurs also in the corpus - this part might be called *positive representativity*
- the representativity wrt. *the absence of all invalid bigrams* of the language in the corpus, which means that if any bigram [First,Second] is a bigram which cannot occur in a correct (i.e. grammatical) sentence of the language, then such a bigram does not occur in the corpus - this part might be called *negative representativity*.

If a corpus is both positively and negatively representative, then indeed it can be said to be a qualitatively representative corpus[5]. In our particular example this means that a bigram occurs in a qualitatively representative (wrt. bigrams) corpus if and only if it is a possible bigram in the language (and from this it already follows that any unigram occurs in such a corpus if and only if it is a possible unigram[6]). From this formulation, it is also clear that the qualitative representativity depends on the notion of grammaticality, that is, on the "language competence" – on the ability of distinguishing between a grammatical and an ungrammatical sentence.

The *quantitative representativity* of a corpus wrt. bigrams can then be approximated as the requirement that the frequency of any bigram and any unigram occurring in the corpus be in the proportion "as in the language performance" to the frequency of occurrence of all other bigrams or unigrams, respectively[7]. However, even when its basic idea is quite intuitive and natural, it is not entirely clear whether quantitative representativity can be formalized rigorously. At stake is measuring the occurrence of a bigram (and of a unigram) within the "complete language performance", understood as set of utterances of a language. This set, however, is infinite if considered theoretically (i.e. as set of all possible utterances in the language) and finite but practically unattainable if considered as a set of utterances realized within a certain time span (also, due to immanent language change, it is questionable whether the concept of set of utterances over a time span is a true performance of a single language). Notwithstanding these problems, the frequencies are used in practice (e.g., for the purpose of training statistical taggers), and hence it is useful to state openly what they really mean: in our example, it is the relative frequencies of the bigrams (and unigrams) in a particular (learning or otherwise referential) corpus. For this reason, since we would not like to be bound to a particular corpus, we refrain from quantitative representativity in the following and we shall deal only with qualitative representativity.

3 PoS-Tagging Errors Detection

In this (core) section, we shall concentrate on methods and techniques of generating "almost error-free" corpora, or, more precisely, on the possibilities of (semi-)automatic detection (and hence correction) of errors in a PoS-tagged corpus. Due to this, i.e. to the aim of achieving an "error-free" corpus, we shall not distinguish between errors due to incorrect tagging, faulty conversion or ill-formed input, and we shall treat them on a par.

The approach as well as its impact on the correctness of the resulting corpus will be demonstrated on the version 2 of the NEGRA corpus of German (for the corpus itself see www.coli.uni-sb.de/sfb378/negra-corpus, for description cf. (Skut et al. 1997)). However, we believe the solutions developed and presented in this paper are not bound particularly to correcting this corpus or to German, but hold generally.

The error search we use has several phases which differ in the amount of context that has to be taken into consideration during the error detection process. Put plainly, the extent of context mirrors the linguistic complexity of the detection, or, in other words, at the moment when the objective is to search for "complex" errors, the "simple(r)" errors should be already eliminated. The first, preliminary phase, is thus the search for errors which are detectable in the minimal local context of one neighbouring word.

3.1 Bootstrapping Impossible Bigrams

Our starting point is the search for "impossible bigrams". These as a rule occur in a realistic large-scale PoS-tagged corpus, for the following reasons:

- in a hand tagged corpus, an "impossible bigram" results from (and unmistakeably signals) either an ill-formed text in the corpus body (including wrong conversion) or a human error in tagging
- in a corpus tagged by a statistical tagger, an "impossible bigram" may result also from an ill-formed source text, as above, and further either from incorrect tagging of the training data (i.e. the error was seen as a "correct configuration (bigram)" in the training data, and was hence learned by the tagger) or from the process of so-called "smoothing", i.e. of assignment of non-zero probabilities also to configurations (bigrams, in the case discussed) which were not seen in the learning phase[8].

For learning the process of detecting errors in PoS-tagging, let us make a provisional and in practice unrealistic assumption (which we shall correct immediately) that we have a qualitatively representative (wrt. bigrams) corpus of sentences of a certain language at our disposal.

Given such a (hypothetical) corpus, all the bigrams in the corpus are to be collected to a set **CB** (correct bigrams), and then the complement of **CB** to the set of all possible bigrams is to be computed; let this set be called **IB** (incorrect bigrams). The idea is now that if any element of

[5] The definitions of positive and negative representativity are obviously easily transferable to cases with other definitions of a phenomenon. Following this, the definition of qualitative representativity holds of course generally, not only in the particular case of a corpus representative wrt. bigrams.

[6] This assertion holds only on condition that each sentence of the language is of length two (measured in words) or longer. Similarly, a corpus qualitatively representative wrt. trigrams is qualitatively representative wrt. bigrams and wrt. unigrams only on condition that each sentence is of length three at least, etc.

[7] From this it easily follows that any quantitatively representative corpus is also a qualitatively representative corpus.

[8] This "smoothing" is necessary in any purely statistical tagger since - put very simply - otherwise configurations (bigrams) which were not seen during the learning phase cannot be processed if they occur in the text to be tagged.

IB occurs in a PoS-tagged corpus whose correctness is to be checked, then the two adjacent corpus positions where this happened must contain an error (which then can be corrected).

When implementing this approach to error detection, it is first of all necessary to realize that learning the "impossible bigrams" is extremely sensible to both aspects of the qualitative representativity of the learning corpus:

- *the lack of negative representativity:* The presence of an erroneous bigram in the set of **CB** causes that the respective error cannot be detected in the corpus whose correctness is to be checked (even a single occurrence of a bigram in the learning corpus means correctness of the bigram),
- *the lack of positive representativity:* The absence of a correct bigram from the **CB** set causes this bigram to occur in **IB**, and hence any of its occurrences in the checked corpus to be marked as a possible error (absence of a bigram in the learning corpus means incorrectness of the bigram).

However, the available corpora are[9] - at least as a rule - not (qualitatively) representative. Therefore, in practice this deficiency has to be compensated for by appropriate means. When applying the approach to NEGRA, we employed

- bootstrapping for achieving positive representativity as good as possible on a given "training" corpus
- manual pruning of the **CB** and **IB** sets for achieving negative representativity.

We started by very careful hand-cleaning errors in a very small sub-corpus of about 80 sentences (about 1.200 words). From this small corpus, we generated the **CB** set, and pruned it manually, using linguistic knowledge (as well as linguistic imagination) about German syntax. Based on the **CB** set achieved, we generated the corresponding **IB** set and pruned it manually again. The resulting **IB** set was then used for automatic detection of "suspect spots" in the sample of next 500 sentences from the corpus, and for hand-elimination of errors in this sample where appropriate (obviously, not all **IB** violations were genuine errors !). Thus we arrived at a cleaned sample of 580 sentences, which we used just in the same way for generating **CB** set, pruning it, generating **IB** set and pruning this set, arriving at an **IB** set which we used for detection of errors in the whole body of the corpus (about 20.500 sentences, 350.000 positions).

The procedure was then re-applied to the whole corpus. For this purpose, we divided the corpus into four parts of approximately 5.000 sentences each. Then, proceeding in four rounds, first the **IB** set was generated (without manual checking) out of 15.000 sentences and then the **IB** set was applied to the rest of the corpus (on the respective 5.000-sentence partition). The corrections based on the results improved the corpus to such an extent that we made the final round, this time dividing the corpus into 20 partitions with approximately 1.000 sentences each and then reapplying the whole process 20 times.

[9] and will hardly ever become, disregarding their size: e.g., in the body of the 100.000.000 positions of the Czech National Corpus, we easily dicovered a case of a missing trigram (and there are most probably many more missing - we just did not search for them)

3.2 Bootstrapping Impossible *n*-grams

The "impossible bigrams" are a powerful tool for checking the correctness of a corpus, however, a tool which works on a very local scale only, since it is able to detect solely errors which are detectable as deviations from the set of possible pairs of adjacently standing tags. Thus, obviously, quite a number of errors remain undetected by such a strategy. As an example of such an as yet "undetectable" error in German we might take the configuration where two words tagged as finite verbs are separated from each other by a string consisting of nouns, adjectives, articles and prepositions only. In particular, such a configuration is erroneous since the rules of German orthography require that some kind of clause separator (comma, dash, coordinating conjunction) occur inbetween two finite verbs[10].

In order to be able to detect also such kind of errors, the above "impossible bigrams" have to be extended substantially. Searching for the generalization needed, it is first of all necessary to get a linguistic view on the "impossible bigrams", in other words, to get a deeper insight into the impossibility for a certain pair of PoS-tags to occur immediately following each other in any linguistically correct and correctly tagged sentence. The point is that this indeed does not happen by chance, that any "impossible bigram" comes into being as a violation of a certain - predominantly syntactic[11] - rule(s) of the language. Viewed in more detail, these violations might be of the following nature:

- *violation of constituency:* The occurrence of an "impossible bigram" in the text signals that - if the tagging were correct - there is a basic constituency relation violated (resulting in the occurrence of the "impossible bigram"); as an example of such configuration, we might consider the bigram PREPOSITION - FINITE VERB (possible German example string: *...für-PREP reiche-VFIN...*). From this it follows that either there is indeed an error in the source text (in our example, probably a missing word, e.g., *Der Sprecher der UNO-Hilfsorganisation teilte mit, für Arme reiche diese Hilfe nicht.*) or there was a tagging error detected (in the example, e.g., an error as in the sentence *... für reiche Leute ist solche Hilfe nicht nötig...*). The source of the error is in both cases violation of the linguistic rule postulating that, in German, a preposition must always be followed by

[10] At stake are true regular finite forms, exempted are words occurring in fixed collocations which do not function as heads of clauses. As an example of such usage of a finite verb form, one might take the collocation *wie folgt*, e.g., in the sentence *Diese Übersicht sieht wie folgt aus:* ... Mind that in this sentence, the verb *folgt* has no subject, which is impossible with any active finite verb form of a German verb subcategorizing for a subject (and possible only marginally with passive forms, e.g., in *Gestern wurde getanzt*, or – obviously – with verbs which do not subcategorize for a subject, such as *frieren, grauen* in *Mich friert, Mir graut vor Statistik*).

[11] Examples of other such violations are rare and are related mainly to phonological rules. In English, relevant cases would be the word pairs *an table, a apple*, provided the tagset were so fine-grained to express such a distinction, better examples are to be found in other languages, e.g. the case of the Czech ambiguous word *se*, cf. (Oliva to appear).

a corresponding noun (NP) or at least by an adjectival remnant of this NP[12].

- *violation of feature cooccurrence rules* (such as agreement, subcategorization, etc.): The point here is that there exist configurations such that if two wordforms (words with certain morphological features) occur next to each other, they necessarily stand in such a configuration, and because of this also in a certain grammatical relation. This relation, in turn, poses further requirements on the (morphological) features of the two wordforms, and if these requirements are not met, the tags of the two wordforms result in an "impossible bigram". Let us take an example again, this time with tags expressing also morphological characteristics: if the words ... *Staaten schickt* ... are tagged as *Staaten-NOUN-MASC-PL-NOM* and *schickt-MAINVERB-PRES-ACT-SG*, then the respective tags *NOUN-MASC-PL-NOM* and *MAINVERB-PRES-ACT-SG* (in this order) create an "impossible bigram". The reason for this bigram being impossible is that if a noun in nominative case occurs in a German clause headed by a finite main verb different from *sein/werden* (which, however, are not tagged as main verbs in the STTS tagset used in NEGRA), then either this noun must be the verb's subject, which in turn requires that the noun and the verb agree in number, or that the noun is a part of coordinated subject, in which case the verb must be in plural. The configuration from the example meets neither of these conditions, and hence it generates an "impossible bigram".

The central observation lies then in the fact that the property of being an impossible configuration can often be retained also after the components of the "impossible bigram" get separated by material occurring inbetween them. Thus, for example, in both our examples the property of being an impossible configuration is conserved if an adverb is placed inbetween, creating thus an "impossible trigram". In particular, in the first example, the configuration *PREP ADV VFIN* cannot be a valid trigram, exactly for the same reasons as *PREP VFIN* was not a valid bigram: *ADV* is not a valid NP remnant. In the second case, the configuration *NOUN-MASC-PL-NOM ADV MAINVERB-PRES-ACT-SG* is not a valid trigram either, since obviously the presence (or absence) of an adverb in the sentence does not change the subject-verb relation in the sentence. In fact, due to recursivity of language, also two, three and in fact any number of adverbs would not make the configuration grammatical and hence would not disturb the error detection potential of the "extended impossible bigrams" from the examples.

These linguistic considerations have a straightforward practical application. Provided a qualitatively representative (in the above ideal sense) corpus is available, it is possible to construct the **IB** set. Then, for each bigram *[First,Second]* from this set, it is possible to collect all trigrams of the form *[First,Between,Second]* occurring in the corpus, and collect all the possible tags *Between* in the set *Possible_Inner_Tags*. Furthermore, given the impossible bigram *[First,Second]* and the respective set *Possible_Inner_Tags*, the learning corpus is to be searched for all tetragrams *[First,Middle_1,Middle_2, Second]*. In case one of the tags *Middle_1, Middle_2* occurs already in the set *Possible_Inner_Tags*, no action is to be taken, but in case the set *Possible_Inner_Tags* contains neither of *Middle_1, Middle_2*, both the tags *Middle_1* and *Middle_2* are to be added into the set *Possible_Inner_Tags*. The same action is then to be repeated for pentagrams, hexagrams, etc., until the maximal length of sentence in the learn corpus prevents any further prolongation of the *n*-grams and the process terminates.

If now the set *Impossible_Inner_Tags* is constructed as the complement of *Possible_Inner_Tags* relatively to the whole tagset, then any *n*-gram consisting of the tag *First*, of any number of tags from the set *Impossible_Inner_Tags* and finally from the tag *Second* is very likely to be an *n*-gram impossible in the language and hence if it occurs in the corpus whose correctness is to be checked, it is to be signalled as a "suspect spot". Obviously, this idea is again based on the assumption of qualitative representativity of the learning corpus, so that for training on a realistic corpus the correctness of the resulting "impossible *n*-grams" has to be hand-checked. This, however, is well-worth the effort, since the resulting "impossible *n*-grams" are an extremely efficient tool for error detection. The implementation of the idea is a straightforward extension of the above approach to "impossible bigrams". The respective algorithm in a semi-formal coating looks like as in Fig 1.

The above approach does not guarantee, however, that all "impossible *n*-grams" are considered. In particular, any "impossible trigram" *[First,Second,Third]* cannot be detected as such (i.e. as impossible) if the *[First,Second]*, *[Second,Third]* and *[First,Third]* are all possible bigrams (i.e. they all belong to the set **CB**). Such an "impossible trigram" in German is, e.g., *[nominative-noun, main_verb,nominative-noun]* - this trigram is impossible[13] since no German verb apart from *sein/werden* (which, as said above, are not tagged as main verbs in NEGRA) can occur in a context where a nominative noun stands both to its right and to its left, however, all the respective bigrams occur quite commonly (e.g., *Johann schläft, Jetzt schläft Johann, König Johann schläft*). Here, an obvious generalization of the approach from "impossible bigrams" to "impossible trigrams" (and "impossible tetragrams", etc.) is possible, however, we did not perform this in full due to the amount of possible trigrams as well as to the data sparseness problem which, taken together, would make the manual work on checking the results unfeasible in practice. We rather applied only about 20 "impossible trigrams" and 6 "impossible tetragrams" stemming from "linguistic invention" (such as the trigram discussed above).As above, this empirical (performance-based) result has to be checked manually (through a human language competence) for correctness, since the performance results might be distorted by tagging errors or by lack of representativity of the corpus.

[12] unlike English, (standard) German has no preposition stranding and similar phenomena - we disregard the colloquial examples like *Da weiss ich nix von.*

[13] Exempted are quotations and other metalinguistic contexts, such as *Der Fluss heisst Donau, Peter übersetzte Faust - eine Tragödie ins Englische als Fist - one tragedy,* which, however, are as a rule lexically specific and hence can be coped with as such.

```
forall invalid_bigram [First, Second]
{ n := 3;
  possible_i_t := Ø;
  while n =< maximal_sentence_length_in_corpus
        do { find all inner-sentential n-grams [First, V1, V2, .., Vn-2, Second];
             for each n-gram found
                 do if {V1, V2, .., Vn-2} ∩ possible _i_t = Ø
                    then allowed_i_t := possible _i_t U {V1, V2, .., Vn-2};
             n := n + 1;
           };
      impossible_i_t([First, Second]) := tagset - possible _i_t;
}
```

Figure 1: Algorithm for Bootstrapping Negative *n*-grams

The above approach does not guarantee, however, that all "impossible *n*-grams" are considered. For example, any "impossible trigram" *[First,Second,Third]* cannot be detected as such (i.e. as impossible) if the *[First,Second]*, *[Second,Third]* and *[First,Third]* are all possible bigrams (i.e. they all belong to the set **CB**). Such an "impossible trigram" in German is, e.g., *[nominative-noun, main_verb,nominative-noun]* - this trigram is impossible[14] since no German verb apart from *sein/werden* (which, as said above, are not tagged as main verbs via the STTS tagset used in NEGRA) can occur in a context where a nominative noun stands both to its right and to its left, however, all the respective bigrams occur quite commonly (e.g., *Johann schläft, Jetzt schläft Johann, König Johann schläft*). Here, an obvious generalization of the approach from "impossible bigrams" to "impossible trigrams" (and "impossible tetragrams", etc.) is possible, however, we did not perform this in full due to the amount of possible trigrams as well as to the data sparseness problem which, taken together, would make the manual work on checking the results unfeasible in practice. We rather applied only about 20 "impossible trigrams" and 6 "impossible tetragrams" stemming from "linguistic invention" (such as the trigram discussed above).

4 Evaluation of the Results

By means of the error-detection techniques described above, we were able to correct 2.661 errors in the NEGRA corpus. These errors were of all sorts mentioned in Sect. 1, however the prevailing part was that of incorrect tagging (only less than 8% were genuine source errors, about 26% were errors in segmentation). The whole resulted in changes on 3.774 lines of the corpus; the rectification of errors in segmentation resulted in reducing the number of corpus positions by over 700, from 355.096 to 354.354

After finishing the corrections, we experimented with training and testing the TnT tagger (Brants, 2000) on the "old" and on the "corrected" version of NEGRA. We used the same testing as described by Brants, i.e. dividing each of the corpus into ten contiguous parts of equal size, each part having parallel starting and end position in each of the versions, and then running the system ten times, each time training on nine parts and testing on the tenth part, and finally computing the mean of the quality results. In doing so, we arrived at the following results:

- if both the training and the testing was performed on the "old" NEGRA, the tags assigned by the TnT tagger differed from the hand-assigned tags within the test sections on (together) 11.138 positions (out of the total of 355.096), which yields the error rate of 3,14%
- if both the training and the testing was performed on the "correct" NEGRA, the tags assigned by the TnT tagger differed from the hand-assigned tags of the test sections on (together) 10.889 positions (out of the total of 354.354), which yields the error rate of 3,07%
- in the most interesting final experiment, the training was performed on the "old" and the testing on the "correct" NEGRA; in the result, the tags assigned by TnT differed from the hand-assigned tags in the test sections on (together) 12.075 positions (out of the total of 354.354), yielding the error rate of 3,41%.

These results show that there was only a negligible (and, according to the χ^2 test, statistically insignificant) difference between the results in the cases when the tagger was both trained and tested on "old" corpus and both trained and tested on the "corrected" corpus. However, the difference in the error rate when the tagger was once trained on the "old" and once on the "corrected" version, and then in both cases tested on the "corrected" version[15], brought up a relative error improvement of 9,97%. This improvement documents the old and hardly surprising truth that - apart from the size - also the correctness of the training data is absolutely essential for the results of a statistical tagger.

Conclusions

The main contribution of this paper lies in the presentation of a method for detecting errors in part-of-speech tagged corpus which is both quite powerful (as to coverage of errors) and - due to bootstrapping - easy to apply, and hence it offers a relatively low-cost means for achieving high-quality PoS-tagged corpora. The main advantage is that the approach described is based on the combination of focussed search for errors of a particular, specific type with bootstrapping of the search, which makes it possible to detect errors even in a very large corpus where manual checking would not be feasible (at least in practice), since it requires passing through the

[14] Exempted are quotations and other metalinguistic contexts, such as *Der Fluss heisst Donau, Peter übersetzte Faust - eine Tragödie ins Englische als Fist - one tragedy,* which, however, are as a rule lexically specific and hence can be coped with as such.

[15] For obvious reasons, we did not even consider training on the "corrected" corpus and testing on the "old" one.

whole of the text and paying attention to all kinds of possible violations - while the approach described concentrates on violations of particular phenomena on particular spots. Hence, it allows for straightforward checking whether an error really occurs - and if so, for a direct correction.
As a side-effect, it should be also mentioned that the method allows not for detecting errors only, but also for detecting inconsistencies in hand-tagging (i.e. differences in application of a given tagging scheme by different human annotators and/or in different time), and even inconsistencies in the tagging guidelines. A particular issue is further the area of detecting and tagging idioms and collocations, in the particular case when these take a form which makes them deviate from the rules of standard syntax (i.e. they are detected as "suspect spots" by the method). For details on all these points, including the particular problems encountered in NEGRA, cf. (Květoň and Oliva in prep.).

Acknowledgement

This work has been sponsored by the *Fonds zur Förderung der wissenschaftlichen Forschung (FWF), Grant No. P12920.* The *Austrian Research Institute for Artificial Intelligence (ÖFAI)* is supported by the *Austrian Federal Ministry of Education, Science and Culture.*

References

Brants T. (2000). *TnT – A Statistical part-of-speech tagger*, in: Proceedings of the 6th Applied Natural Language Processing conference, Seattle

Hirakawa H., Ono K. and Yoshimura Y. (2000). *Automatic refinement of a PoS tagger using a reliable parser and plain text corpora*, in: Proceedings of the 18th Coling conference, Saarbrücken

Květoň P. and Oliva K. (in prep.). *Correcting the NEGRA Corpus: Methods, Results, Implications*, ÖFAI Technical Report

Müller F.H. and Ule T. (2001). *Satzklammer annotieren und tags korrigieren: Ein mehrstufiges top-down-bottom-up System zur flachen, robusten Annotierung von Sätzen im Deutschen*, in: Proceedings der GLDV-Frühjahrstagung 2001, Gießen

NEGRA. www.coli.uni-sb.de/sfb378/negra-corpus

Oliva K. (2001). *The possibilities of automatic detection/correction of errors in tagged corpora: a pilot study on a German corpus*, in: 4th International conference "Text, Speech and Dialogue" TSD 2001, Lecture Notes in Artificial Intelligence 2166, Springer, Berlin 2001

Oliva K. (to appear). *Linguistics-based tagging of Czech: disambiguation of 'se' as a test case*, in: Proceedings of 4th European Conference on Formal Description of Slavic Languages held in Potsdam from 28th till 30th November 2001

Petkevič V. (2001). *Grammatical agreement and automatic morphological disambiguation of inflectional languages*, in: 4th International conference "Text, Speech and Dialogue" TSD 2001, Lecture Notes in Artificial Intelligence 2166, Springer, Berlin 2001

Schiller A., Teufel S., Stöckert C. and Thielen C. (1999). *Guidelines für das Tagging deutscher Text corpora*, University of Stuttgart / University of Tübingen

Skut W., Krenn B., Brants T. and Uszkoreit H. (1997). *An annotation scheme for free word order languages*, in: Proceedings of the 3rd Applied Natural Language Processing Conference, Washington D.C.

A Comparison Of Efficacy And Assumptions Of Bootstrapping Algorithms For Training Information Extraction Systems

Rayid Ghani[*] **and Rosie Jones**[†]

[*]Accenture Technology Labs
Chicago, IL 60601, USA
rayid.ghani@accenture.com

[†]School of Computer Science
Carnegie Mellon University, Pittsburgh PA 15213, USA
rosie.jones@cs.cmu.edu

Abstract

Information Extraction systems offer a way of automating the discovery of information from text documents. Research and commercial systems use considerable training data to learn dictionaries and patterns to use for extraction. Learning to extract useful information from text data using only minutes of user time means that we need to leverage unlabeled data to accompany the small amount of labeled data. Several algorithms have been proposed for bootstrapping from very few examples for several text learning tasks but no systematic effort has been made to apply all of them to information extraction tasks. In this paper we compare a bootstrapping algorithm developed for information extraction, meta-bootstrapping, with two others previously developed or evaluated for document classification; cotraining and coEM. We discuss properties of these algorithms that affect their efficacy for training information extraction systems and evaluate their performance when using scant training data for learning several information extraction tasks. We also discuss the assumptions underlying each algorithm such as that seeds supplied by a user will be present and correct in the data, that noun-phrases and their contexts contain redundant information about the distribution of classes, and that syntactic co-occurrence correlates with semantic similarity. We examine these assumptions by assessing their empirical validity across several data sets and information extraction tasks.

1. Introduction

Information Extraction systems offer a way of automating the discovery of information from text documents. Both research and commercial systems for information extraction need large amounts of labeled training data to learn dictionaries and extraction patterns. Collecting these labeled examples can be very expensive, thus emphasizing the need for algorithms that can provide accurate classifications with only a a few labeled examples. One way to reduce the amount of labeled data required is to develop algorithms that can learn effectively from a small number of labeled examples augmented with a large number of unlabeled examples.

Several algorithms have been proposed for bootstrapping from very few examples for several text learning tasks. Using Expectation Maximization to estimate maximum a posteriori parameters of a generative model for text classification (Nigam et al., 2000), using a generative model built from unlabeled data to perform discriminative classification (Jaakkola and Haussler, 1999), and using transductive inference for support vector machines to optimize performance on a specific test set (Joachims, 1999) are some examples that have shown that unlabeled data can significantly improve classification performance, especially with sparse labeled training data. For information extraction, Yangarber et al. used seed information extraction template patterns to find target sentences from unlabeled documents, then assumed strongly correlated patterns are also relevant, for learning new templates. They used an unlabeled corpus of 5,000 to 10,000 documents, and suggest extending the size of the corpus used, as many initial patterns are very infrequently occurring (Yangarber et al., 2000a; Yangarber et al., 2000b).

A related set of research uses labeled and unlabeled data in problem domains where the features naturally divide into two disjoint sets. Blum and Mitchell (Blum and Mitchell, 1998) presented an algorithm for classifying web pages that builds two classifiers: one over the words that appear on the page, and another over the words appearing in hyperlinks pointing to that page. Datasets whose features naturally partition into two sets, and algorithms that use this division, fall into the co-training setting (Blum and Mitchell, 1998). Meta-Bootstrapping (Riloff and Jones, 1999) is an approach to learning dictionaries for information extraction starting only from a handful of phrases which are examples of the target class. It makes use of the fact that noun-phrases and the partial-sentences they are embedded in can be used as two complementary sources of information about semantic classes. Similar methods have been used for named entity classification (Collins and Singer, 1999).

Although a lot of effort has been devoted to developing bootstrapping algorithms for text learning tasks, there has been very little work in systematically applying these algorithms for information extraction and evaluating them on a common set of documents. All of the previously mentioned techniques have been tested on different types of problems, with different sets of documents, under different experimental conditions, thus making it difficult to objectively evaluate the applicability and effectiveness of these algorithms. In this paper, we first describe a range of bootstrapping approaches that fall into the cotraining setting and lay out the underlying assumptions for each. We then experimentally compare the performance of each algorithm on a common set of information extraction tasks and documents and relate it to the degree to which the assumptions are satisfied in the data sets and semantic learning tasks.

2. The Information Extraction Task

The information extraction tasks we tackle in this paper involve extracting noun phrases that fall into the following three semantic classes: organizations, people and locations. It is important to note that although named entity recognizers are usually used to extract these classes, the distinction we make in this paper is to extract all noun phrases (including "construction company", "jail warden", and "far-flung ports") instead of restricting our task to only proper nouns (which is the case in standard named entity recognizers). Because our focus is extraction of general semantic classes, we have not used many of the features common in English-language named entity recognition, including ones based on sequences of charactes in upper case, and matches to dictionaries, though adding these could improve the accuracy for these classes. This is important to note since that makes it likely that our results will translate to other semantic classes which are not found in online lists or written in capital letters.

The techniques we compare here are similar to those that have been used for semantic lexicon induction (eg (Riloff and Jones, 1999)). However, we believe that the noun-phrases we extract should be taken "in context". Thus, terms we generally consider unambiguous, such as place-names or dictionary terms, can now have different meanings depending on the context that they occur in. For example, the word "Phoenix" usually refers to a location, as in the following sentence:

A scenic drive from Phoenix lies a place of legendary beauty.

but can also refer to the "Phoenix Land Company", as in this sentence:

Phoenix seeks to divest non-strategic properties if alternate uses cannot de monstrate sustainable 20% returns on capital investment.

We can group these types of occurences in three broad categories:

General Polysemy: many words have multiple meanings. For example, "company" can refer to a commercial entity or to companionship.

General Terms: many words have a broad meaning that can refer to entities of various types. For example, "customer" can refer to a person or a company.

Proper Name Ambiguity: proper names can be associated with entities of different types. For example, "John Hancock" can refer to a person or a company, sicne companies are often named after people.

In general, we belive that the context determines whether the meaning of the word can be further determined and that we can correctly classify the noun phrase into the semantic class by examining the immediate context, in addition to the words in the noun phrase. Therefore we approach this problem as an information extraction task, where the goal is to extract and label noun phrase instances that correspond to semantic categories of interest.

3. Data Set and Representation

As our data set, we used 4392 corporate web pages collected for the WebKB project (Craven et al., 1998) of which 4160 were used for training and 232 were set aside as a test set. We preprocessed the web pages by removing HTML tags and adding periods to the end of sentences when necessary.[1] We then parsed the web pages using a shallow parser.

We marked up the held out test data by labeling each noun phrase as one or more of (NP) instance as an organization, person, location, or none. We addressed each task as a binary classification task. Each *noun phrase context* consists of two items: (1) the noun phrase itself, and (2) and the context (an extraction pattern). We used the AutoSlog (Riloff, 1996) system to generate extraction patterns.

By using both the noun phrases and the contexts surrounding them, we provide two different types of features to our classifier. In many cases, the noun phrase itself will be unambiguous and clearly associated with a semantic category (e.g., "the corporation" will nearly always be an organization). In these cases, the noun phrase alone would be sufficient for correct classification. In other cases, the context itself is a dead give-away. For example, the context containing the pattern "subsidiary of <np>" nearly always extracts an organization. In those cases, the context alone is sufficient. However, we suspect that both the noun phrase and the context often play a role in determining the correct classification.

4. Bootstrapping Algorithms

In this section we give a brief overview of each of the algorithms we will be using for bootstrapping. We analyze how the properties and assumptions of each may affect accuracy.

4.1. Baseline Methods

Since our bootstrapping algorithms all use seed noun-phrases for an initial labeling of the training data, we should look at how much of their accuracy is based on the use of those seeds, and how much is derived from bootstrapping using those seeds. To this end, we implemented two baselines which use *only* the seeds, or noun-phrases containing the seeds, but use no bootstrapping.

4.1.1. Extraction Using Seeds Only

All the algorithms we describe use seeds as their source of information about the target class. A useful way of assessing what we gain by using a bootstrapping algorithm is to use the seeds as our sole model of information about the target class. The seeds we use for bootstrapping all algorithms are shown in Table 1.

[1]Web pages pose a problem for parsers because separate lines do not always end with a period (e.g., list items and headers). We used several heuristics to insert periods whenever an independent line or phrase was suspected.

The algorithm for seed extraction is: any noun-phrase in the test set exactly matching a word on the seed list is assigned a score of 1. All other noun-phrases are assigned the prior.

4.1.2. Head Labeling Extraction

All the bootstrapping algorithms we discuss use the seeds to perform *head-labeling* to initialize the training set. The algorithm for head labeling is: any noun-phrase in the training set whose head matches a word on the seed list is assigned a score of 1. This may not lead to completely accurate initialization, if any of the seeds are ambiguous. We will discuss this in more detail in Section 5.1.

In order to evaluate the contribution of the head-labeling to overall performance of the bootstrapping, we performed experiments using the head-labeling alone as information in order to extracted from the unseen test set.

The algorithm for *head labeling extraction* is: any noun-phrase in the test set whose head matches a word on the seed list is assigned a score of 1. All other noun-phrases are assigned the prior.

4.2. Bootstrapping Methods

The bootstrapping methods we describe fall under the cotraining setting where the features naturally partition into multiple disjoint sets, any of which individually is sufficient to learn the task. The separation into feature sets we use for the experiments in this paper is that of noun-phrases, and noun-phrase-contexts.

4.2.1. Cotraining

Cotraining (Blum and Mitchell, 1998) is a bootstrapping algorithm that was originally developed for combining labeled and unlabeled data for text classification. At a high level, it uses a feature split in the data and starting from seed examples, labels the unlabeled data and adds the most confidently labeled examples incrementally. When used in our information extraction setting, the algorithm details are as follows:

1. Initialize NPs from both positive and negative seeds
2. Use labeled NPs to score contexts
3. Select k most confident positive and negative contexts, assign them the positive and negative labels
4. Use labeled contexts to label NPs
5. Select k most confident positive and negative NPs, assign them the positive and negative labels
6. goto 2.

Note that cotraining assumes that we can accurately model the data by assigning noun-phrases and contexts to a class. When we add an example, it is either a member of the class (assigned to the positive class, with a probability of 1.0) or not (assigned to the negative class, with a probability of 0.0 of belonging to the target class). As we will see in section 5.2., many noun-phrases, and many more contexts, are inherently ambiguous. Cotraining may harm its performance through its hard (binary 0/1) class assignment.

4.2.2. CoEM

coEM was originally proposed for semi-supervised text classification by Nigam & Ghani (Nigam and Ghani, 2000) and is similar to the cotraining algorithm described above, but incorporates some features of EM. coEM uses the feature split present in the data, like co-training, but is instead of adding examples incrementally, it is iterative, like EM. It starts off using the same initialization as cotraining and creates two classifiers (one using the NPs and the other using the context) to score the unlabeled examples. Instead of assigning the scored examples positive or negative labels, coEM uses the scores associated with *all* the examples and adds *all* of them to the labeled set probabilistically (in the same way EM does for semi-supervised classification). This process iterates until the classifiers converge.

Muslea et al. (Muslea et al., 2000) extended the co-EM algorithm to incorporate active learning and showed that it has a robust behavior on a large spectrum of problems because of its ability to ask for the labels of the most ambiguous examples, which compensates for the weaknesses of the underlying semi-supervised algorithm.

In order to apply coEM to learning information extraction, we seed it with a small list of words. All noun-phrases with those words as heads are assigned to the positive class, to initialize the algorithm.

Note that coEM does not perform a hard clustering of the data, but assigns probabilities between 0 and 1 of each noun-phrase and context belonging to the target class. This may reflect well the inherent ambiguity of many terms.

4.2.3. Meta-bootstrapping

Meta-bootstrapping (Riloff and Jones, 1999) is a simple two-level bootstrapping algorithm using two features sets to label one another in alternation. It is customized for information extraction, using the feature sets *noun-phrases* and *noun-phrase-contexts* (or *caseframes*). There is no notion of negative examples or features, but only positive features and unlabeled features. The two feature sets are used asymmetrically. The noun-phrases are used as initial data and the set of positive features grows as the algorithm runs, while the noun-phrase-contexts are relearned with each outer iteration.

Heuristics are used to score the features from one set at each iteration, based on co-occurrence frequency with positive and unlabeled features, using both frequency of co-occurrence, and diversity of co-occurring features. The highest scoring features are added to the positive feature list.

Meta-bootstrapping treats the noun-phrases and their contexts asymmetrically. Once a context is labeled as positive, *all* of its co-occurring noun-phrases are assumed to be positive. However, a noun-phrase labeled as positive is part of a committee of noun-phrases voting on the next context to be selected. After a phase of bootstrapping, all contexts learned are discarded, and only the best noun-phrases are retained in the permanent dictionary. The bootstrapping is then recommenced using the expanded list of noun-phrases. Once a noun-phrase is added to the permanent dictionary, it is assumed to be representative of the positive class, with confidence of 1.0.

Class	Seeds
locations	australia, canada, china, england, france, germany, japan, mexico, switzerland, united states
organizations	inc., praxair, company, companies, dataram, halter marine group, xerox, arco, rayonier timberlands, puretec
people	customers, subscriber, people, users, shareholders, individuals, clients, leader, director, customer

Table 1: Seeds used for initialization of bootstrapping.

4.3. Active Initialization

As we saw in the discussion of head-labeling (Section 4.1.2.), using seed words for initializing training may lead to initialization that includes errors. We give measures of the rate of errors in head-labeling in Table 3. We will augment the intialization of bootstrapping by correcting those errors before bootstrapping begins, and seeing the effects on test set extraction accuracy. We call this *active initialization*, by analogy to active learning.

5. Assumptions in Bootstrapping Algorithms

The bootstrapping algorithms described in Section 4.2. have a number of assumptions in common; that initialization from seeds leads to labels which are accurate for the target class, that seeds will be present in the data, that similar syntactic distribution correlates with semantic similarity, and that noun-phrases and their contexts are redundant and unambiguous with respect to the semantic classes we are attempting to learn. We assess the validity of each of these assumptions by examining the data.

5.1. Initialization from Seeds Assumption

All the algorithms we describe use seed words as their source of information about the target class. An assumption made by all the algorithms we present is that seed words suggested by a user will be present in the data. We assess this by comparing seed density for three different tasks over two types of data, one collected specifically for the task at hand, one drawn according to a uniform random distribution over documents on the world wide web. The seeds we use for initializing bootstrapping all algorithms are shown in Table 1. We show the density of seed words in different corpora in Table 2. Note that the `people` and `organizations` classes are much more prevalent in the company data we are working with than in documents randomly obtained using Yahoo's random URL page.

Another assumption that arises from using seeds is that labeling using them accurately labels items in the target semantic class. All three algorithms initialize the unlabeled data by using the seeds to perform *head labeling*. Any noun-phrase with a seed word as its head is labeled as positive. For example, when *canada* is in the seed word list, both "eastern canada" and "marketnet inc. canada" are labeled as being positive examples. Table 3 shows the accuracy for `locations` and `people`. For `people`, some words were mostly unambiguous, with the exception of a few examples, "customers", which was unambigous except in prhases such as "industrial customers". The seed-word "people" also led to some training examples of questionable utility, for example "invest in people". If we learn the context "invest in", it may not help in learning to extract words for people, in the general case. Other seed-words from the `people` class proved to be very ambiguous; "leader" was most often to used to describe a company, as in the sentence "Anacomp is a world leader in digital document-management services".

We will discuss the results of correcting these errors before beginning bootstrapping in Section 6.3.

Corpus	Class	Seed-density (/10,000)
fixed	locations	18
random		21
fixed	organizations	112
random		17
fixed	people	70
random		33

Table 2: Density of seed words per 10,000 noun-phrases in fixes corpus of company web pages, and corpus of randomly collected web pages.

Class	Accuracy
`locations`	98%
`people`	95%

Table 3: Accuracy of labeling examples automatically using seed-heads.

5.2. Feature Sets Redundancy Assumption

The bootstrapping algorithms we discuss all assume that there is sufficient information in each feature set (noun-phrases and contexts) to use either to label an example. However, when we look at the ambiguity of noun-phrases in the test set (Table 4) we see that 81 noun-phrases were ambiguous between two classes, and 4 were ambiguous between three classes. This means that these 85 noun-phrases (2% of the 4413 unique noun-phrases occurring in the test set) are not in fact sufficient to identify the class. This discrepancy may hurt cotraining and meta-bootstrapping more, since they assume that we can classify noun-phrases into a class with 100% accuracy.

When we examine the same information for contexts (Table 4) we see even more ambiguity. 36% of contexts are ambiguous between two or more classes.

We have another measure of the inherent ambiguity of the noun-phrases making up our target class when we measure the inter-rater(labeler) agreement on the test set. We randomly sampled 230 examples from the test collection, broken into two subsets of size 114 and 116 examples. We had four labelers label subsets with different amounts of information. The three conditions were:

- noun-phrase, local syntactic context, and full sentence (*all*)
- noun-phrase, local syntactic context (*np-context*)

Ambiguity	**Class(es)**	**Number of NPs**
1	none	3574
	loc	114
	org	451
	person	189
2	loc, none	6
	org, none	31
	person, none	25
	loc, org	6
	org, person	13
3	loc, org, none	1
	org, person, none	3

Table 4: Distribution of test NPs in classes

Ambiguity	**Class(es)**	**Number of Pats**
1	none	1068
	loc	25
	org	98
	person	59
2	loc, none	51
	org, none	271
	person, none	206
	loc, org	5
	org, person	50
3	loc, org, none	18
	org, person, none	83
4	loc, org, person, none	6

Table 5: Distribution of test patterns in classes

- noun-phrase only (*np*).

The labelers were asked to label each example with any or all of the labels `organization`, `person` and `location`. Before-hand, they each labeled 100 examples separate from those described above (in the *all* condition) and discussed ways of resolving ambiguous cases (agreeing, for example, to count "we" as both `person` and `organization` when it could be referring to the organization or the individuals in it. The distribution of conditions to labelers is shown in Figure 6.

We found that when the labelers had access to the noun-phrase, context, and the full sentence they occurred in, they agreed on the labeling 90.5% of the time. However, when one did not have the sentence (only the noun-phrase and context), agreement dropped to 88.5%. Our algorithms have only the noun-phrase and contexts to use for learning. Based on the agreement of our human labelers, we conjecture that the algorithms could do better with more information.

Labeler	Set 1 Condition	Set 2 Condition
1	NP-context	all
2	all	NP-context
3	NP	all
4	all	NP

Table 6: Conditions for inter-rate evaluation - All stands for NP, context and the entire sentence in which the NP-context pair appeared

5.3. Syntactic - Semantic Correlation Assumption

All the algorithms we address in this paper use the assumption that phrases with similar syntactic distributions have similar semantic meanings. It has been shown (Dagan et al., 1999) that syntactic cooccurrence leads to clusterings which are useful for natural language tasks. However, since we seek to extract items from a single semantic target class at a time, syntactic correlation may not be sufficient to represent our desired semantic similarity.

The mismatch between syntactic correlation and semantic similarity can be measured directly by measuring context ambiguity, as we did in Section 5.2.. Consider the context "visit <X>", which is ambiguous between all four of our classes `location`, `person`, `organization` and `none`. It occurs as a `location` in "visit our area", ambiguously between `person` and `organization` in "visit us", and as `none` in "visit our website".

Similarly, examining the ambiguous noun-phrases we see that occurring with a particular noun-phrase does not necessarily determine the semantics of a context. Three of the three-way ambiguous noun-phrases in our test set are: "group", "them" and "they". Adding "they" to the model when learning one class may cause an algorithm to add contexts which belong to a different class.

Meta-bootstrapping deals with this problem by specifically forbidding a list of 35 stop words (mainly prepositions) from being added to the dictionaries. In addition, the heuristic that a caseframe be selected by many different noun-phrases in the seed list helps prevent the addition of a single ambiguous noun-phrase to have too strong an influence on the bootstrapping. The probabilistic labeling used by coEM helps prevent problems from this ambiguity. Though we also implemented a stop-list for cotraining, its all-or-nothing labeling means that ambiguous words not on the stop list (such as "group") may have a strong influence on the bootstrapping.

6. Empirical Comparison of Bootstrapping Algorithms

After running bootstrapping with each algorithm we have two models: (1) a set of noun-phrases, with associated probabilities or scores, and (2) a set of contexts with probabilities or scores. We then use these models to extract examples of the target class from a held-out hand annotated test corpus. Since we are able to associate scores with each test example, we can sort the test results by score, and calculate precision-recall curves.

6.1. Extraction on the Test Corpus

There are several ways of using the models produced by bootstrapping to extract from the test corpus:

1. Use only the noun-phrases. This corresponds to using bootstrapping to acquire a lexicon of terms, along with probabilities or weights reflecting confidence assigned by the bootstrapping algorithm. This may have advantage over lists of terms (such as proper names) which

have no such probabilities associated with them. The probabilities allow us to sort extracted phrases and thus control whether we obtain few, highly probable members of the target class, or obtain good coverage, at the expense of accuracy. We will measure these trade-offs using precision and recall, discussed in Section 6.2..

2. Use only the contexts. In this case we discard the noun-phrases we learned during bootstrapping, and use only the contexts as extraction patterns for extracting on the test set. We extract a noun-phrase when it occurs with one of the contexts in our model, using the score assigned by that context. This may have the advantage of allowing greater generalization. Unseen words and phrases can be extracted from the test corpus, and overspecialization based on the training corpus can be avoided.

3. Use both models. To score a noun-phrase context pair in the test set, assume independence, and multiply the model noun-phrase and context scores to get a probability for the example. Noun-phrases and contexts not seen in the training corpus are given a score based on the prior probability. This has the advantage of combining all the information we acquired during training. This method is most effective for methods which assign probability-like scores (coEM and cotraining). For meta-bootstrapping, there is no natural way of combining the scores.

We experimented with these extraction methods for all three algorithms, and found that method 2, extracting using only the contexts, was by far the best for meta-bootstrapping, so all our results for meta-bootstrapping use this extraction method. CoEM and cotraining performed best with method 3, combining information from both noun-phrase and context models, so all results reported for coEM and cotraining use this extraction method.

6.2. Evaluation

We use the models to score all noun-phrase instances in the test corpus, using context-scoring for meta-bootstrapping, and noun-phrase-context scoring for coEM and cotraining, as described in Section 6.1.. Since we could select a variety of thresholds if we used our models for classifiation, depending on the target application, we use a large number of thresholds, calculating precision and recall for each. Precision is given by

$$Precision = \frac{tp_t}{tp_t + fp_t}$$

where tp_t is the number of correct examples above the threshold, and fp_t is the number of incorrect examples above the threshold. Recall is given by

$$Recall = \frac{tp_t}{tp_t + fn_t}$$

where tp_t is the number of correct examples above the threshold and fn_t is the number of correct examples below the threshold.

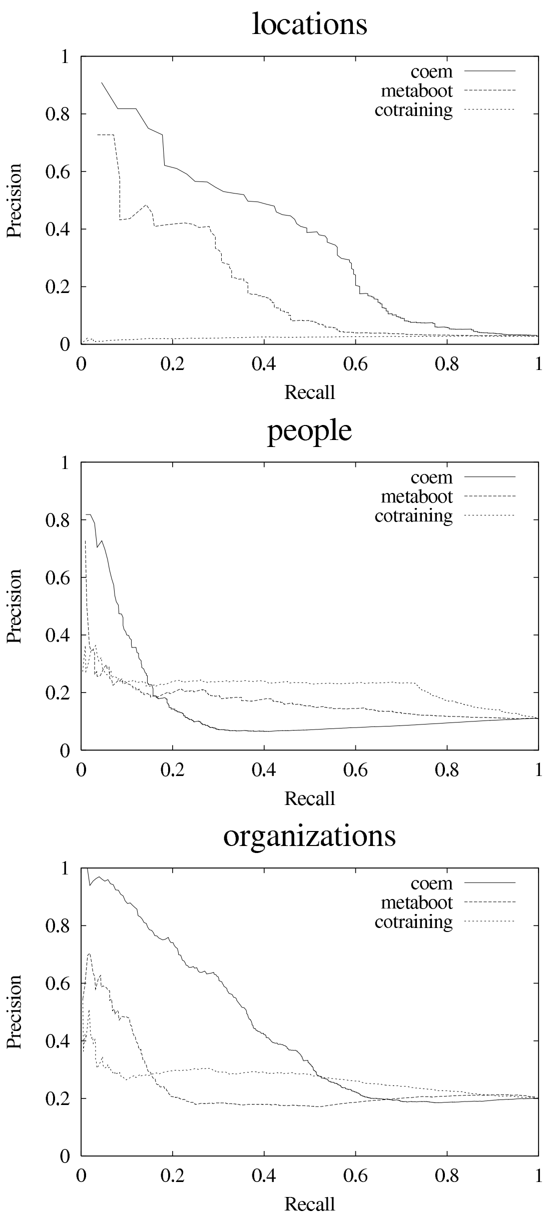


Figure 1: Comparison of bootstrapping using coEM, meta-bootstrapping and cotraining, for the classes `locations`, `people` and `organizations`.

6.3. Experimental Results

Figure 1 compares using models obtained by bootstrapping with coEM, meta-bootstrapping and cotraining, for extracting on a held out test set. CoEM performs better than meta-bootstrapping, while cotraining does very poorly.

Figure 2 shows that bootstrapping using unlabeled documents gives us significant gains over using just the seeds, or noun-phrases with the seeds as heads, for extracting from the test corpus. This difference is least marked for the class `people`, which had the most ambiguous seed words.

Figure 3 shows that only a small gain is obtained by hand-labeling all 669 examples matching the `location` seeds before commencing bootstrapping, and all 2521 examples matching the `people` class before commencing bootstrapping.

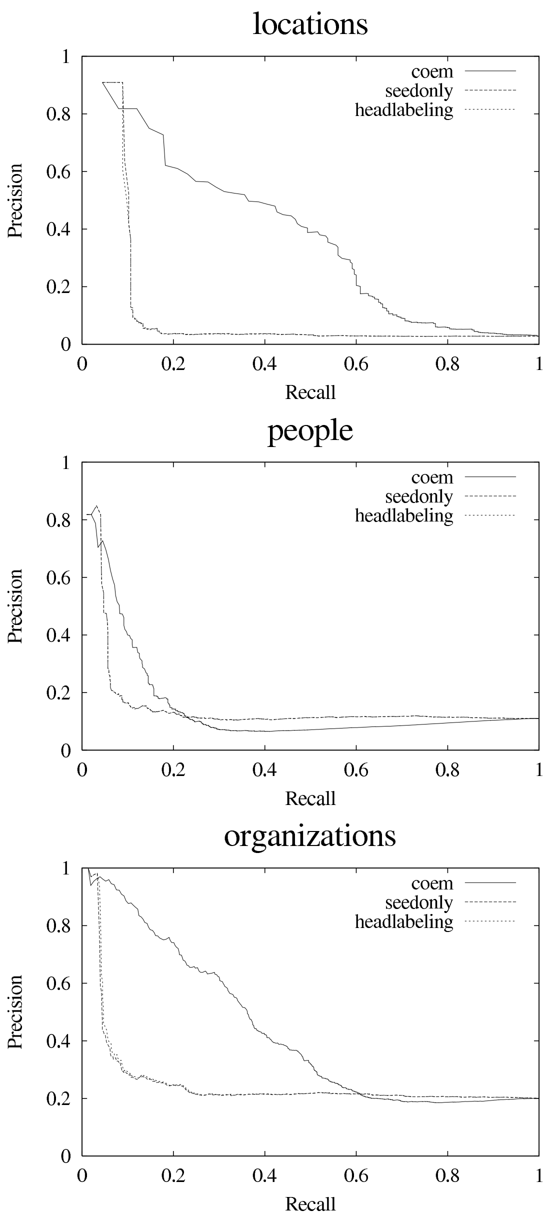


Figure 2: Comparison of the effects of using seeds alone, noun-phrases with seeds as heads (head-labeling) and models learned by bootstrapping with coEM to extract on the unseen test set. Seeds and head-labeling lead to good precision, but poor recall. Bootstrapping using coEM improves recall without loss of precision.

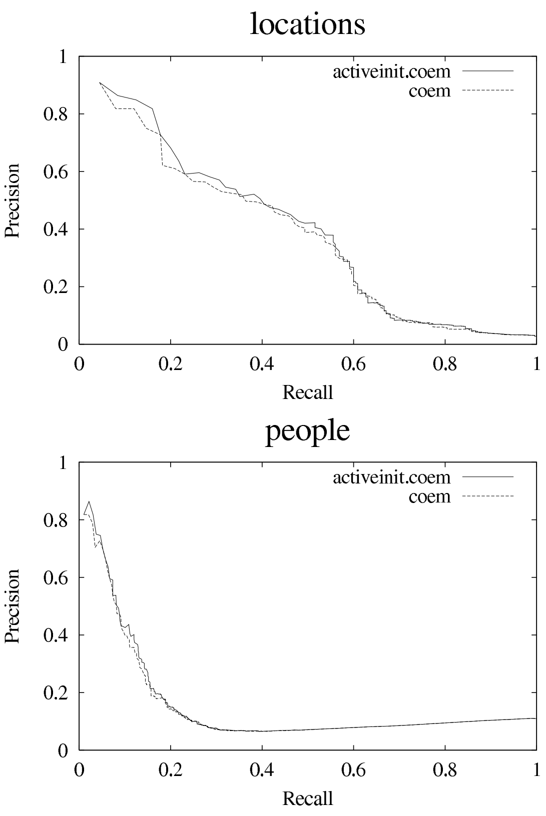


Figure 3: Comparison of the effects of hand-labeling all examples matching the seed-words before commencing bootstrapping (active initialization), against bootstrapping assuming all are correct (coem). A small gain is obtained by labeling all data input.

7. Discussion

The advantage coEM has over meta-bootstrapping and cotraining may reflect the good match between its probabilistic treatment of the data, and the inherent ambiguity of the classes. This permits an ambiguous example to be labeled with a probability that reflects its true ambiguity, rather than committing it to a class, then being overly influenced by its presence in that class. Since meta-bootstrapping repeatedly discards the contexts, ambiguity in the contexts does not hurt the algorithm as much as it hurts cotraining.

We can see from the comparison of gains from bootstrapping over using the seeds or head-labeling, that classes for which we have ambiguous seeds words, such as our `people` class benefit less from bootstrapping than those with relatively unambiguous seed words. However, we still benefit from bootstrapping. This may be because the noise introduced by the ambiguous seed-words is somewhat mitigated by the presence of the less ambiguous seed words.

For `locations` and `people` we saw that correcting by hand the examples labeled using the seed words did not have a significant impact on the results. This means that for relatively unambiguous seed words, at least, hand-labeling them in context does not give us an advantage over using automatic head-labeling.

For the seed-words and datasets we used, seed density in the training corpus does not appear to be a major issue.

8. Conclusions and Future Work

We presented a range of bootstrapping algorithms for information extraction and provide experimenal results comparing cotraining, coEM and meta-bootstrapping over a common set of documents and semantic learning tasks. We also analyzed the underlying assumptions for each of the algorithms and found that performance is affected by the degree to which the assumptions are violated in the data set and the task at hand.

We also analyzed several ways of initializing the bootstrapping algorithms and found that the accuracy does not appear to hinge greatly on initialization that is 100% accurate. A greater density of seeds in the training set for a class (`organizations` and `people` had greater seed density than `locations`) does not appear to lead to greater extraction accuracy on the held out test set. Algorithms which cater to the ambiguity inherent in the feature set are more reliable for bootstrapping, whether they do that by using the feature sets asymmetrically (like meta-bootstrapping), or by allowing probabilistic labeling of examples (like coEM).

Although we have limited the scope of this paper to algorithms that utilize a feature split present in the data (co-training setting), we believe that this comparison of algorithms should be extended to settings where such a split of the features dies not exist, for examples algorithms like expectation maximization (EM) over the entire combined feature set. It would also be helpful to extend the analysis to a greater variety of semantic classes and larger sets of documents.

Acknowledgements

We thank Tom Mitchell and Ellen Riloff for numerous, extremely helpful discussions and suggestions that contributed to the work described in this paper.

9. References

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

M. Collins and Y. Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1999. Similarity-based models of cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69.

Tommi Jaakkola and David Haussler. 1999. Exploiting generative models in discriminative classifiers. In *Advances in NIPS 11*.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of ICML '99*.

Ion Muslea, Steven Minton, and Craig A. Knoblock. 2000. Selective sampling with redundant views. In *AAAI/IAAI*, pages 621–626.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pages 86–93.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

Ellen Riloff and Rosie Jones. 1999. Learning Dictionaries for Information Extraction by Multi-level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press.

E. Riloff. 1996. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. 85:101–134.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000a. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*.

R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000b. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing, (ANLP-NAACL 2000)*, pages 282–289.

Using Decision Trees to Predict Human Nouns in Spanish Parsed Text

Marisa Jiménez

Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
marialj@microsoft.com

Abstract

This paper discusses the use of decision tree models in the acquisition of human nouns for a Spanish monolingual dictionary. This method uses decision tree models to learn the contexts in which a pre-classified set of human nouns occur. We then use the predictions of the model to acquire new human nouns at run time during sentence parsing. The acquisition process was done in 5 stages. First, we annotated automatically all nouns in a selected corpus from Spanish Encarta as "human" and "non-human". Then, we parsed all sentences in the corpus, and extracted linguistic features from the parsed sentences in which each annotated noun occurs. Afterwards we built decision tree models using the data and the features extracted. The task was to classify and assign probabilities to the contexts in which human nouns occur. Finally, we dynamically acquired new human nouns for the Spanish dictionary during sentence parsing; we used the predictions made by the model in this acquisition task.

1. Introduction

Manual annotation schemes to acquire lexical knowledge are costly and time-consuming. To circumvent this problem, different methods to bootstrap already annotated data have been proposed in the literature. One of the bootstrapping methods proposed is using already existing taggers to annotate more data. Some of the work reported focuses on the use of already existing taggers to create mappings between the older tagger and the new tag set (Atwell et al., 1994; Teufel, 1994, among others). Other work proposes combining existing taggers to improve accuracy rates (Van Halteren et al., 1998; Brill and Wu, 1998; Van Halteren et al., 2000; Zavrel and Daelemans, 2000).

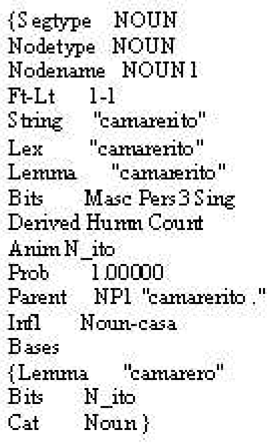
Another bootstrapping method that has been proposed is using the statistical distributions of already lexically-classified words to classify new words (Stevenson et al., 1999; Stevenson and Merlo, 1997; Schulte im Walde, 1998). Stevenson and Merlo (2000) discuss a method to automatically classify verbs into semantic classes by looking at the statistical distributions of a few annotated verbs within a big corpus.

Along the lines of the second method mentioned above, this paper discusses a bootstrapping technique to acquire human nouns for a Spanish monolingual dictionary. This method uses decision tree models to learn the contexts in parsed text in which a pre-classified set of human nouns occur. The predictions of the model are then used in the acquisition of new human nouns at run time during sentence parsing.

2. Human Nouns in our Spanish NLP System

The Spanish monolingual dictionary that is part of our Natural Language Processing (NLP) system contains 140,664 entries, of which 72,445 are nouns. Out of these 72,445 nouns, 9,068 are tagged as human nouns in the dictionary. These human nouns were annotated partially by hand and also automatically by using information in their dictionary definitions.

Our system has several strategies to deal with human nouns occurring in text and that are not part of the Spanish dictionary. The first strategy is using derivational morphology rules. If a noun is not in the Spanish dictionary, the system tries to derive it morphologically from a noun that is present in the dictionary. Furthermore, if the noun from which this unfound noun is derived is human, we copy the human information from the base noun. In figure 1 we provide an example of a noun record created derivationally. The noun *camarerito* 'little waiter' is derived from *camarero* 'waiter'. As the base noun is tagged Humn (which stands for "human") in the dictionary, the human tag is copied to the derived noun as well.



```
{Segtype   NOUN
Nodetype   NOUN
Nodename   NOUN1
Ft-Lt     1-1
String    "camarerito"
Lex       "camarerito"
Lemma      "camarerito"
Bits      Masc Pers3 Sing
Derived Humn Count
Anim N_ito
Prob      1.00000
Parent    NP1 "camarerito ."
Infl      Noun-casa
Bases
{Lemma     "camarero"
Bits      N_ito
Cat       Noun}
```

Figure 1. Example of a human noun record created by derivational morphology

We also have a strategy to identify human names that are not in the dictionary when they occur in a collocation. In figure 2, we provide an example of a human name identified by our system, *Boris Karloff.* Although neither *Boris* nor *Karloff* is in the Spanish dictionary, the system is able to recognize them as the first name and last name of a person. Nevertheless, if either *Boris* or *Karloff* appears alone, they are not identified as human names.

```
{Segtype   NOUN
 Nodetype  NOUN
 Nodename  NOUN1
 Ft-Lt     1-2
 String    "Boris Karloff"
 CopyOf    NOUN2
 Rules     (GatherNames)
 Constits  (NOUN3 NOUN2)
 Lemma     "Boris_Karloff"
 Bits      Masc Pers3 Sing PrprN
           Factoid InitCap Humn
           Nme Anim Fnme Unfnd
 Prob      1.00000
 Parent    NP1 "Boris Karloff."
 Factrecs  FIRSTNAME1 "Boris"
           LASTNAME1 "Karloff"
 FactPred  person
 FactClass PERSON }
```

Figure 2. Record of a human name identified by our system

Despite these strategies, our system sometimes fails to identify some human nouns that are encountered in text. Knowing whether a noun is human is essential for our Spanish parser as this information is used to identify sentential subjects. Sentential position alone is not sufficient for successful subject identification because Spanish subjects may appear in multiple positions.

Decisions on subject identification are taken as our parser builds up the syntactic tree. Whether a noun is human or not is crucial for subject identification in many instances. One of these cases is when a sentence contains two noun phrases (NPs) that both appear to the right of the verb. If one of the NPs is recognized as human, and the other is not, our parser takes the human NP to be the subject of the sentence.

```
Ayer declaró la ley marcial el presidente de la república.
DECL1     AVP1      ADV1*     "Ayer"
          VERB1*    "declaró"  (Subject NP1  Object NP2)
          NP2       DETP1     ADJ1*     "la"
                    NOUN1*    "ley"  (Predadj AJP1)
                    AJP1      ADJ2*     "marcial"
          NP1       DETP2     ADJ3*     "el"
                    NOUN2*    "presidente"
                    PP1       PP2       PREP1*    "de"
                              DETP3     ADJ4*     "la"
                              NOUN3*    "república"
          CHAR1     "."
```

Translation: Yesterday, the president of the Republic declared martial law.

Figure 3. Example of a Spanish sentence with two NPs to the right of the verb.

In figure 3 we provide an example of a Spanish sentence where human information on an NP is used for subject identification. In this sentence there are two NPs appearing to right of the verb *declaró* 'you-formal declared', NP1, *el presidente de la república* 'the president of the Republic', and NP2, *la ley marcial* 'martial law'. In order to determine that NP1 is the subject of the sentence, the parser uses the fact that the head of the NP *presidente* is marked human in the Spanish dictionary.

3. Motivation and Experiment Design

Motivated by the importance of human nouns for our NLP system, we designed a bootstrapping method to add new human nouns to the Spanish dictionary. This method uses decision tree models to learn the contexts in parsed text in which a pre-classified set of human nouns occur. The predictions made by the model are then used in the dynamic acquisition of new human nouns during sentence parsing.

There were 5 stages in the experiment design:

- Automatic annotation of all nouns in a selected corpus into "human" and "non-human".
- Parsing of sentences in the selected corpus.
- Linguistic feature extraction from the parsed sentences in which each annotated noun occurs. The goal was determining which features were relevant or not with respect to human nouns.
- Building decision tree models using the features extracted. The task was to classify and assign probabilities to the contexts in which human nouns occur.
- Dynamically adding new human nouns to the Spanish dictionary based on the predictions made by the model.

In section 4 we will describe the first 4 stages of our experiment, which have to do with the different steps in building the decision tree models. In section 5 we will discuss the dynamic acquisition of new human nouns using the model predictions.

4. Using Decision Trees to Predict Human Nouns

4.1. Data and Feature Extraction

We used the Spanish version of Encarta as the data resource for our experiment because this encyclopedia is a good source of human nouns. We gathered 126,935 sentences, and extracted all their nouns. There were a total of 641,673 nouns, which we then annotated automatically. Those nouns that were recognized as human by our system were tagged as "human", and the rest were tagged as "not-human". Unfound words were excluded from the annotation task for obvious reasons.

We were quite confident that these automatic tags had a high degree of accuracy. Our confidence was based on the fact that the Spanish system has mechanisms to identify human nouns that are not in the dictionary. Furthermore, over the years we have done manual revisions of the 10,000 most common nouns in the Spanish dictionary; in these revisions we made sure that all the human nouns in the high-frequency set were tagged correctly.

Despite our degree of confidence in the accuracy of the automatic tags, we did some manual revision to have an estimation of our error rate. We reviewed 3,000 tagged nouns by hand; they were extracted at random from different parts of the corpus. We did not find any errors in the subset of tags reviewed; this gave us confidence that the error rate was small.[1]

[1] We would like to thank an anonymous reviewer for his/her comments on making sure that our tags were correct. His/her

The next step in the experiment was parsing all the sentences associated with the tagged nouns. We discarded sentences that did not obtain a complete parse. Afterwards, we automatically extracted 232 linguistic features from the parsed tree of the sentence associated with each tagged noun. Our approach was extracting the full set of features available in the parse, instead of performing manual feature selection. Nevertheless, one of the features extracted, "the ending of the noun", was selected manually. We included this morphological feature because of its highly suspected relevance.

The pool of extracted features fell into the following categories:

- All verbal features present in the main verb of the sentence.
- Selected features of the parent and the grandparent of the noun, such as gender, number, and (in) definiteness, among others.
- All the features present in the pre-modifiers and post-modifiers of the parent.
- The lemma of the preposition governing the parent, if present.
- The syntactic label of the parent and the grandparent.
- The ending of the noun.

In figure 4 we provide a sample of some features extracted for the noun *películas* 'movies'. The noun and its sentence are listed first. Under *Values* some values for the features extracted are listed. The first value is always the value of the tag, which is "NoHuman" in this case.

Sentence: [películas]: Durante los últimos veinticinco años de su vida, Gabin hizo unas veinticuatro películas, entre ellas destaca El clan de los sicilianos (1969) de Henri Verneuil, obra maestra del cine negro francés.

Translation: During the last twenty-five years of his life, Gabin made twenty-four movies, among them 'The Sicilian Clan' (1969), directed by Henri Verneuil, master piece of the French *film noir*.

Values:
A~is_human_noun = "NoHuman"
A~has_possible_human_ending = "NoPossHumEnding"
1~Sing~Parent = "0"
1~Plur~Parent = "1"
1~Art~Parent = "1"
1~Def~Parent = "0"
1~PPobj~Parent = "0"
1~Nodetype~Parent = "NP"

Figure 4. Values of some of the features extracted for the noun *películas* 'movies'

4.2. Building and Examining the Models

Once the feature extraction was completed, we used the data and the values extracted to build decision tree models. The goal was to classify the features by their relevance in predicting human nouns. Decision trees are a popular tool used in machine-learning for classification tasks. They provide a classification of selected features and rank their relative importance in predicting a target feature. The tools that we used to build our decision trees are the WinMine toolkit (Chickering *et al.*, 1997, n.d.), developed at Microsoft Research. Decision trees built by WinMine predict a probability distribution over all possible target values.

These tools take as input a text file with the characteristics shown in figure 4. The data is split into training and testing at a 70/30 rate. For both training and testing, we only extracted features from sentences that had a complete parse. The tools produce several decision tree models at different levels of accuracy. All the models are in xml format.

In order to inspect the models, we use a model viewer that allows the user to view the shape of the decision tree. This tool also shows the relative importance of all the relevant features selected by the model. All features that are found relevant in the human model are connected with arrows to the target feature that we are trying to predict, which appears in the center of the viewer. The features that are not found relevant by the model are shown disconnected at the bottom of the screen.

In figure 5 we provide a snapshot of the four best predictors in the model for human nouns. The viewer has a slider (to the left of the figure) that highlights the strongest predictors as the slider goes up. The following characteristics of the main verb are the top predictors in the model shown in figure 5:

- The verb takes a nominal predicate object (e.g.: *Lo considero mi amigo* 'I consider him my friend').
- The verb is transitive.
- The verb is transitive and followed by preposition + infinitive (e.g.: *Te animo a venir* 'I encourage you to come').
- The verb takes an adjectival predicate object (e.g.: *No te creo tonto* 'I don't believe you stupid').

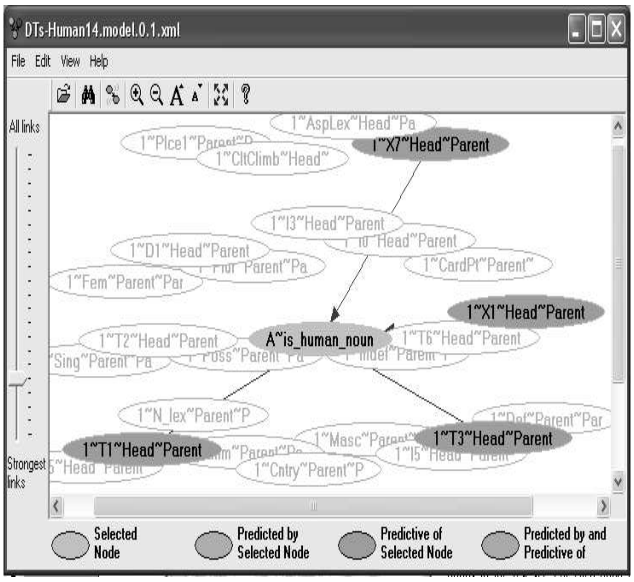


Figure 5. Snapshot of the four top predictors in the model for human nouns

concern was that we would tag as "non-human" true human nouns that were just missing the tag in our dictionary. The manual revision of a 3000 noun sample confirmed to us that the error rate was small.

4.3. Model Predictions

Out of the 232 features that were extracted for each noun, 111 were found to have predictive value in the model for human nouns. The selected features fell into the following categories:

- Ending of the noun. Certain endings such as *–or* '-er', and *–ista* '-ist' were found to be strong indicators that the noun is human.
- Governing preposition of the parent. Certain prepositions such as *a* 'to' and *por* 'by' were found likely to govern a human noun, while others such as *en* 'in' tend not to govern human nouns.
- The noun was upper case. Upper case was found to be a strong predictor in certain contexts, for example, nominal apposition.
- Syntactic label of the parent and grandparent.
- Whether the parent is in apposition to another noun.
- Syntactic features of the sentence main verb.
- Whether the parent of the noun has pre-modifiers.
- Some features of the pre-modifiers and post-modifiers of the parent (e.g.: whether the parent has a definite post-modifier, and the pre-modifier is a possessive pronoun).
- Whether the parent is definite.
- Whether the parent has a post-modifier that is a relative clause.

Some of the predictions made by the model were pretty straightforward, such as the relevance of the noun ending or capitalization. Other predictions were not as straightforward, such as the relevance of the parent being post-modified by a relative clause.

One of the main advantages of the decision tree model for human nouns was its complexity. The model had 1,194 branching nodes, which means that 1,194 linguistic decisions were made when predicting whether a noun is likely to be human or not. Manually coding each one of these decisions would be extremely time-consuming.

As for evaluation numbers, the best model had an overall accuracy of 84.29% over a 69.83% baseline. The baseline corresponds to the accuracy if the most frequent value (non-human) had been assigned to all nouns in the test set. In table 1 we provide the evaluation numbers for this model. These calculations are based on the 1922502 nouns in the test set. For each noun in the test set, the value predicted by the model was compared to the value observed.

	Human	Non-Human	Total
Total	58066	134436	192502
Predicted	54238	138264	
Correctly Predicted	46030	116228	162258
Precision	84.86%	84.06%	
Recall	79.27%	86.45%	
F-measure	81.97%	85.23%	
Baseline			69.83%
Overall Accuracy			84.29%

Table 1. Evaluation numbers for the human model

5. Using the Decision Tree Model to Acquire New Human Nouns

The last step in our experiment was using the predictions made by the model in the acquisition of human nouns that were not yet in the Spanish dictionary. The plan was to add new human noun records dynamically during sentence analysis using the decision tree predictions.

Our NLP system has in place rules to do lexical learning at run time during sentence parsing. Lexical-learning rules are used dynamically to create domain-specific corpus-based lexicons. These domain-specific dictionaries are used as supplements to the general dictionary (see Pentheroudakis (technical report), and Wu et al.,2002).

In preparation for lexical learning, we gathered 57,397 sentences from Spanish Encarta; we made sure that these sentences contained unfound words.[2] Afterwards, the decision tree model for human nouns was invoked from the lexical-learning rules while parsing the sentences.

After parsing the sentences, a new learned lexicon was created; this lexicon contained 16,902 human nouns. After quickly inspecting a handful of dictionary entries, we realized that a good amount of the learned nouns were human names. This realization seemed consistent with the fact that Encarta is an encyclopedia, and that our Spanish dictionary does not contain many proper names. Some examples of the names that were learned were *Filippo, Yourcenar* and *Hemingway*, among others. Among the "non-proper name" human nouns that were learned were *zapatistas* 'zapatists', *antirreeleccionista* 'anti-reelectionist', and *clarinero* 'clarinist', among others.

To evaluate the accuracy of the learned dictionary, we randomly gathered 600 nouns from the pool of sentences used for the creation of the dictionary. We then manually reviewed all the words in the 600 set. We checked whether each noun was truly human or not, and whether it was part of the learned dictionary. 41 nouns from the 600 set were discarded because they were either typos or did not have a noun part of speech; we ended up with a total of 559 nouns.[3] In table 2, we provide the results of the manual evaluation.

	Human	Non-Human	Total
Total	269	290	559
Predicted	230	329	
Correctly Predicted	205	265	470
Precision	89.13%	80.54%	
Recall	76.20%	91.37%	
Baseline			51.88%
F-measure	82.16%	85.61%	
Overall Accuracy			84.07%

Table 2. Summary of manual evaluation of 559 nouns from Encarta

[2] The first time that we invoked the model during parsing, we did not make sure that unfound words were in the data. As a result, few new human nouns were learned.

[3] Our NLP system assigns by default a noun part of speech to unfound words. It is possible that some of the unfound nouns that were in the test set were not nouns.

6. Conclusions

In this paper we have presented a method to augment the number of human nouns in the Spanish dictionary of our NLP system. This method uses decision tree models to learn the contexts in which a pre-classified set of human nouns occur. The experiment was done in 5 stages. First, we annotated automatically all nouns in a selected corpus from Spanish Encarta into "human" and "non-human". We then parsed all sentences in the corpus, and extracted several linguistic features from each parsed sentence. We then built decision tree models using the features extracted. The task was to classify and assign probabilities to the contexts in which human nouns occur. Finally we added dynamically new human nouns to the Spanish dictionary based on the predictions made by the model.

Evaluation of our experiment showed that we were able to learn a significant amount of human nouns at good accuracy levels. This method reduces human effort in dictionary maintenance. We see another two advantages to using decision tree models for human noun acquisition. The first one is that the model makes complex decisions that would be very costly and time-consuming to be hand-coded. And, second, the model made more complete predictions than our native speaker intuitions.

7. Acknowledgments

We would to thank the members of the NLP group at Microsoft Research for their comments and help at various stages during the development of this paper.

8. References

Atwell, E., J. Hughes, and C. Souter (1994). Amalgam: Automatic Mapping among Lexico-grammatical Annotation Models. Technical report, Internal Paper, CCALAS, Leeds University.

Brill, E. and J. Wu (1998). Classifier Combination for Improved Lexical Disambiguation. In *COLING-ACL'98* Montreal, Canada.

Chikering, D. Max nd. *WinMine Toolkit Home Page*. http.//research.microsoft.com/~dmax/WinMine/Tooldoc .htm.

Pentheroudakis, J. (2001). *Lex Rules!*. Technical report.

Schulte im Walde, S., 1998. Automatic Semantic Classification of Verbs according to their Alternation Behaviour. AIMS Report 4(3), IMS, Universität Stuttgart.

Stevenson, S. and P. Merlo (1997). Lexical Structure and Processing Complexity. *Language and Cognitive Processes*, 12(1-2):349-399.

Stevenson, S., P. Merlo, N. Karaeva, and K. Whitehouse (1999). Supervised Learning of Lexical Semantic Verb Classes using Frequency Distributions. In *Procs of SigLex '99*, College Park, Maryland.

Stevenson, S. and P. Merlo (2000). Automatic Lexical Acquisition Based on Statistical Distributions. *Proceedings of COLING 2000*, Saarbrüecken, Germany.

Teufel, S. (1995). A Support Tool for Tagset Mapping. In *Proc. of of the Workshop SIGDAT (EACL95)*

Van Halteren, H., J. Zavrel, and W. Daelemans (1998). Improving Data Driven Wordclass Tagging by System Combination. In *Proceedings of ACL-COLING'98*, Montreal, Canada.

Van Halteren, H., J. Zavrel, and W. Daelemans (2001). Improving Accuracy in NLP through Combination of Machine Learning Systems. *Computational Linguistics 27 (2), 1*99-230.

Wu, A., J. Pentheroudakis, and Z. Jiang (2002). Dynamic Lexical Acquisition in Chinese Sentence Analysis. Submitted to COLING 2002 for consideration.

Zavrel, H.J. and W. Daelemans (2000). Bootstrapping a Tagged Corpus through Combination of Existing Heterogeneous Taggers. *International Conference on Language Resources and Evaluation.* Athens, Greece.

X-TRACTOR: A Tool For Extracting Discourse Markers

Laura Alonso[*], **Irene Castellón**[*], **Lluís Padró**[†]

[*]Department of General Linguistics
Universitat de Barcelona
{lalonso, castel}@lingua.fil.ub.es

[†]TALP Research Center
Software Department
Universitat Politècnica de Catalunya
padro@lsi.upc.es

Abstract

Discourse Markers (DMs) are among the most popular clues for capturing discourse structure for NLP applications. However, they suffer from inconsistency and uneven coverage. In this paper we present X-TRACTOR, a language-independant system for automatically extracting DMs from plain text. Seeking low processing cost and wide applicability, we have tried to remain independent of any hand-crafted resources, including annotated corpora or NLP tools. Results of an application to Spanish point that this system succeeds in finding new DMs in corpus and ranking them according to their likelihood as DMs. Moreover, due to its modular architecture, X-TRACTOR evidences the specific contribution of each out of a number of parameters to characterise DMs. Therefore, this tool can be used not only for obtaining DM lexicons for heterogeneous purposes, but also for empirically delimiting the concept of DM.

1. Motivation

The problem of capturing discourse structure for complex NLP tasks has often been addressed by exploiting surface clues that can yield a partial structure of discourse (Marcu, 1997; Dale and Knott, 1995; Kim et al., 2000). Cue phrases such as *because*, *although* or *in that case*, usually called Discourse Markers (DMs), are among the most popular of these clues because they are both highly informative of discourse structure and have a very low processing cost.

However, they present two main shortcomings: inconsistency in their characterisation and uneven coverage. The lack of consensus about the concept of DM, both theoretically and for NLP applications, is the main cause for these two shortcomings. In this paper, we will show how a knowledge-poor approach to lexical acquisition is useful for addressing both these problems and providing partial solutions to them.

1.1. Delimitation of the concept of DM

A general consensus has not been achieved about the concept of DM. The set of DMs in a language is not delimited, nor by intension neither by extension. But however controversial DM characterisation may be, there is a core of well-defined, prototypical DMs upon which a high consensus can be found in the literature. By studying this lexicon and the behaviour of the lexical units it stores in naturally occurring text, DM characterising features can be discovered. These features can be applied to corpus to obtain lexical items that are similar to the original ones. Applying bootstraping techniques, these newly identified lexical items can be incorporated to the lexicon and this enhanced lexicon can be used for discovering new characterising features. This process can be repeated until the obtained lexical items are not considered valid any more.

It may be argued that enlarging this starting set implies making it more controversial, by adding items whose status as DMs is questionable. However, being empirically grounded, this enlargement is relatively unbiased, and it yields an enhancement of the concept of DM that may be useful for NLP applications.

Taking it to the extreme, unendlessly enhancing the concept of DM implies that anything loosely signalling discourse structure would be considered as a DM. Although this might sound absolutely undesirable, it could be argued that a number of lexical items can be assigned a varying degree of marking strength or *markerhood*[1]. It would be then up to the human expert to determine the load of *markerhood* required for a lexical item to be considered a DM in a determined theoretical framework or application. Lexical acquisition can evidence the load of discursive information in every DM by evaluating it according to the DM characterising features used for extraction.

1.2. Scalability and Portability of DM Resources

Work concerning DMs has been mainly theoretical, and applications to NLP have been mainly oriented to restricted NLGeneration applications. So, DM resources of wide coverage have still to be built. The usual approach to building DM resources is fully manual. For example, DM lexicons are built by gathering and describing DMs from corpus or literature on the subject, a very costly and time-consuming process. Moreover, due to variability among humans, DM lexicons tend to suffer from inconsistency in their extension and intension. To inherent human variability, one must add the general lack of consensus about the appropriate characterisation of DMs for NLP. All this prevents reusability of these costly resources.

[1]By analogy with *termhood*(Kageura and Umino, 1996), which is the term used in terminology extraction to indicate the likelihood that a term candidate is an actual term, we have called *markerhood* the likelihood that a DM candidate is an actual DM.

As a result of the fact that DM resources are built manually, they present uneven coverage of the actual DMs in corpus. More concretely, when working on previously unseen text, it is quite probable that it contains DMs that are not in a manually built DM lexicon. This is a general shortcoming of all knowledge that has to be obtained from corpus, but it becomes more critical with DMs, since they are very sparse in comparison to other kinds of corpus-derived knowledge, such as terminology. As follows, due to the limitations of humans, a lexicon built by mere manual corpus observation will cover a very small number of all possible DMs.

The rest of the paper is organised as follows. In Section 2., we present the architecture of the proposed extraction system, X-TRACTOR, with examples of an application of this system to acquiring a DM lexicon for discourse-based automated text summarisation in Spanish. In Section 2 we present the results obtained for this application, to finish with conclusions and future directions.

2. Proposed Architecture

One of the main aims of this system is to be useful for a variety of tasks or languages. Therefore, we have tried to remain independent of any hand-crafted resources, including annotated texts or NLP tools. Following the line of (Engehard and Pantera, 1994), syntactical information is worked by way of patterns of function words, which are finite and therefore listable. This makes the cost of the system quite low both in terms of processing and human resources.

Focusing on adaptability, the architecture of X-TRACTOR is highly modular. As can be seen in Figure 1, it is based in a language-independent kernel implemented in perl and a number of modules that provide linguistic knowledge.

The input to the system is a starting DM lexicon and a corpus with no linguistic annotation. DM candidates are extracted from corpus by applying linguistic knowledge to it. Two kinds of knowledge can be distinguished: general knowledge from the language and that obtained from a starting DM lexicon.

The DM extraction kernel works in two phases: first, a list of all might-be-DMs in the corpus is obtained, with some characterising features associated to it. A second step consists in ranking DM candidates by their likelihood to be actual markers, or *markerhood*. This ranked list is validated by a human expert, and actual DMs are introduced in the DM lexicon. This enhanced lexicon can be then re-used as input for the system.

In what follows we describe the different parts of X-TRACTOR in detail.

2.1. Linguistic Knowledge

Two kinds of linguistic knowledge are distinguished: general and lexicon-specific. General knowledge is stored in two modules. One of them accounts for the distribution of DMs in naturally occurring text in the form of rules. It is rather language-independant, since it exploits general discursive properties such as the occurrence in discursively salient contexts, like beginning of paragraph or sentence. The second module is a list of stopwords or function words of the language in use.

Lexicon-specific knowledge is obtained from the starting DM lexicon. It also consists of two modules: one containing classes of words that constitute DMs and another with the rules for legally combining these classes of words. We are currently working in an automatic process to induce these rules from the given classes of words and the DMs in the lexicon.

In the application of this system to Spanish, we started with a Spanish DM lexicon consisting of 577 DMs [2]. Since this lexicon is oriented to discourse-based text summarisation, each DM is associated to information useful for the task (see Table 1), such as *rhetoric type*. We adapted the system so that some of this information could also be automatically extracted for the human expert to validate. Results were excellent for the feature of *syntactic type*, and very good for *rhetorical content* and *segment boundary*.

We transformed this lexicon to the kind of knowledge required by X-TRACTOR, and obtained 6 classes of words (adverbs, prepositions, coordinating conjunctions, subordinating conjunctions, pronouns and content words), totalling 603 lexical items, and 102 rules for combining them. For implementation, the words are listed and they are treated by pattern-matching, and the rules are expressed in the form of *if - then - else* conditions on this pattern-matching (see Table 2).

2.2. DM candidate extraction

DM candidates are extracted by applying the above mentioned linguistic knowledge to plain text. Since DMs suffer from data sparseness, it is necessary to work with a huge corpus to obtain a relatively good characterisation of DMs. In the application to Spanish, strings were extracted by at least one of the following conditions:

- Salient location in textual structure: beginning of paragraph, beginning of the sentence, marked by punctuation.
- Words that are typical parts of DMs, such as those having a strong rhetorical content. thetorical content types are similr to those handled in RST (Mann and Thompson, 1988).
- Word patterns, combinations of function words, sometimes also combined with DM-words.

2.3. Assessment of DM-candidate markerood

Once all the possible might-be-DMs are obtained from corpus, they are ponderated as to their *markerhood*, and a ranked list is built.

Different kinds of information are taken into account to assess *markerhood*:

- **Frequency** of occurrence of the DM candidate in corpus, normalised by its length in words and exclusive of stopwords. Normalisation is achieved by the function $normalised\ frequency = length \cdot \log(frequency)$.

[2]We worked with 784 expanded forms corresponding to 577 basic cue phrases

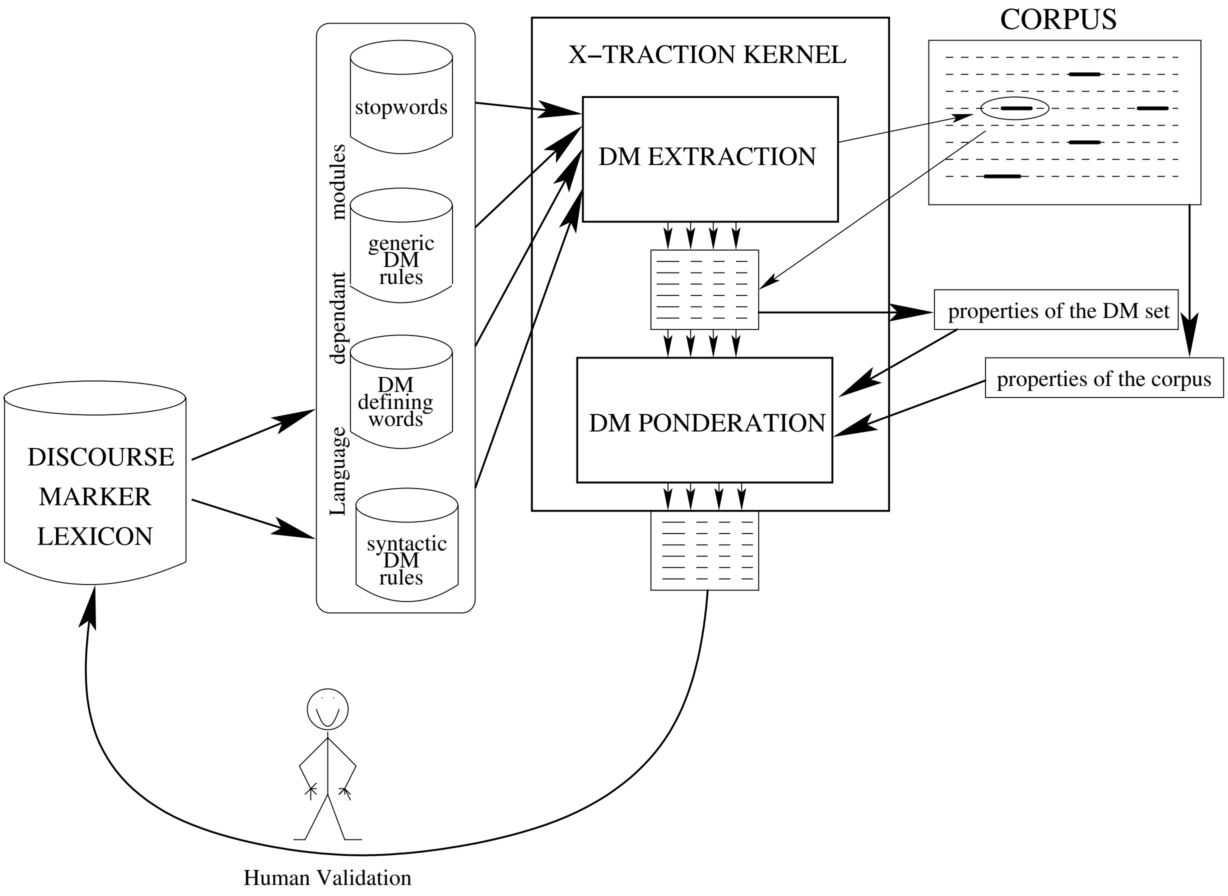


Figure 1: Architecture of X-Tractor

DM	boundary	syntactic type	rhetorical type	direction	con tent
además	not appl.	adverbial	satellizer	inclusion	reinforcement
a pesar de	strong	preposition	satellizer	right	concession
así que	weak	subordinating	chainer	right	consequence
dado que	weak	subordinating	satellizer	right	enablement

Table 1: Sample of the cue phrase lexicon

- Frequency of occurrence in **discursively salient context**. Discursively salient contexts are preferred occurrence locations for DMs. This parameter has been combined with DM classes motivated by clustering in (Alonso et al., 2002).

- **Mutual Information** of the words forming the DM candidate. Word strings with higher mutual information are supposed to be more plausible lexical units.

- **Internal Structure** of the DM, that is to say, whether it follows one of the rules of combination of DM-words. For this application, X-TRACTOR was aimed at obtaining DMs other than those already in the starting lexicon, therefore, longer well-structured DM candidates were priorised, that is to say, the longer the rule that a DM candidate satisfies, the higher the value of this parameter.

- **Rhetorical Content** of the DM candidate is increased by the number of words with strong rhetorical content it contains. These words are listed in one of the modules of external knowledge, and each has a rhetorical content associated to them. This rhetorical content can be pre-assigned to the DM candidate for the human expert to validate.

- **Lexical Weight** accounts for the the presence of non frequent words in the DM candidate. Unfrequent words make a DM with high *markerhood* more likely as a segment boundary marker.

- **Linking Function** of the DM candidate accounts for its power to link spans of text, mostly by reference.

- **Length** of the DM candidate is relevant for obtaining new DMs if we take into consideration the fact that DMs tend to aggregate.

These parameters are combined by weighted voting for *markerhood* assessment, so that the importance of each of them for the final *markerhood* assessment can be adapted

```
for each word in string
    if word is a preposition, then
      if word-1 is an adverb, then
        if word-2 is a coordinating conjunction, then
          if word+1 is a rhetorical-content word, then
            if word+2 is a preposition, then
              assign the DM candidate structural weight 5
            elsif word+2 is a subordinating conjunction, then
              assign the DM candidate structural weight 5
            else assign the DM candidate structural weight 4
          elsif word+1 is a pronoun, then
            assign the DM candidate structural weight 4
        else assign the DM candidate structural weight 3
```

Figure 2: Example of rules for combination of DM-constituing words

to different targets. By assigning a different weight to each one of these parameters, the system can be used for extracting DMs useful for heterogeneous tasks, for example, automated summarisation, anaphora resolution, information extraction, etc.

In the application to Spanish, we were looking for DMs that signal discourse structure useful for automated text summarisation, that is to say, mostly indicators of relevance and coherence relations.

3. Results and Discussion

We ran X-TRACTOR on a sample totalling 350,000 words of Spanish newspaper corpus, and obtained a ranked list of DMs together with information about their syntactical type, rhetorical content and an indication of their potential as segment boundary markers. Only 372 out of the 577 DMs in the DM lexicon could be found in this sample, which indicates that a bigger corpus would provide a better picture of DMs in the language, as will be developed below.

3.1. Evaluation of Results

Evaluation of lexical acquisition systems is a problem still to be solved. Typically, the metrics used are standard IR metrics, namely, *precision* and *recall* of the terms retrieved by an extraction tool evaluated against a document or collection of documents where terms have been identified by human experts (Vivaldi, 2001). Precision accounts for the number of term candidates extracted by the system which have been identified as terms in the corpus, while recall states how many terms in the corpus have been correctly extracted.

This kind of evaluation presents two main problems: first, the bottleneck of hand-tagged data, because a large-scale evaluation implies a costly effort and a long time for manually tagging the evaluation corpus. Secondly, since terms are not well-defined, there is a significant variability between judges, which makes it difficult to evaluate against a sound golden standard.

For the evaluation of DM extraction, these two problems become almost unsolvable. In the first place, DM density in corpus is far lower than term density, which implies that judges should read a huge amount of corpus to identify a number of DMs significant for evaluation. In practical terms, this is almost unaffordable. Moreover, X-TRACTOR's performance is optimised for dealing with huge amounts of corpus. On the other hand, the lack of a reference concept for DM makes inter-judge variability for DM identification even higher than for term identification.

Given these difficulties, we have carried out an alternative evaluation of the presented application of the system. To give a hint of the recall of the obtained DM candidate list, we have found how many of the DMs in the DM lexicon were extracted by X-TRACTOR, and how many of the DM candidates extracted were DMs in the lexicon[3]. To evaluate the goodness of *markerhood* assessment, we have found the ratio of DMs in the lexicon that could be found among the first 100 and 1000 highest ranked DM candidates given by X-TRACTOR. To evaluate the enhancement of the initial set of DMs that was achieved, the 100 highest ranked DMs were manually revised, and we obtained the ratio of actual DMs or strings containing DMs that were not in the DM lexicon. Noise has been calculated as the ratio of non-DMs that can be found among the 100 highest ranked DM candidates.

3.2. Parameter Tuning

To roughly determine which were the parameters more useful for finding the kind of DMs targeted in the presented application, we evaluated the goodness of each single parameter by obtaining the ratio of DMs in the lexicon that could be found within the 100 and 1000 DM candidates ranked highest by that parameter.

In Figure 3 it can be seen that the parameters with best behaviours in isolation are *content*, *structure*, *lexical weight* and *occurrence in pausal context*, although none of them performs above a dummy baseline fed with the same corpus sample. This baseline extracted 1- to 4-word strings after punctuation signs, and ranked them according to their frequency, so that the most frequent were ranked highest. Frequencies of strings were normalised by length, so that $normalised\ frequency = length \cdot \log(frequency)$. Moreover, the frequency of strings containing stopwords was reduced.

[3]We previously checked how many of the DMs in the lexicon could actually be found in corpus, and found that only 386 of them occurred in the 350,000 word sample; this is the upper bound of in-lexicon DM extraction.

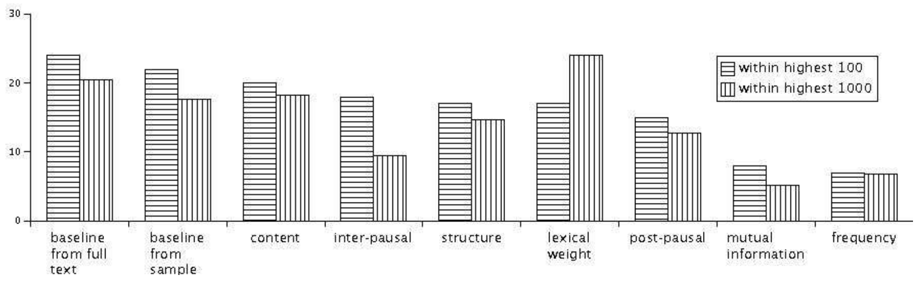


Figure 3: Ratio of DM andidates that contain a DM in the lexicon among the 100 and 1000 highest ranked by each individual parameter

	baseline	X-TRACTOR
Coverage of the DM lexicon	88%	87.5%
ratio of DMs in the lexicon		
within 100 highest ranked	31%	41%
within 1000 highest ranked	21%	21.6%
Noise		
within the 100 highest ranked	57%	32%
Enhancement Ratio		
within the 100 highest ranked	9%	15%

Table 2: Results obtained by X-TRACTOR and the baseline

However, the same dummy baseline performed better when fed with the whole of the newspaper corpus, consisting of 3,5 million words. This, and the bad performance of the parameters that are more dependant on corpus size, like *frequency* and *mutual information*, clearly indicates that the performance of X-TRACTOR, at least for this particular task, will tend to improve when dealing with huge amounts of corpus. This is probably due to the data sparseness that affects DMs.

This evaluation provided a rough intuition of the goodness of each of the parameters, but it failed to capture interactions beteween them. To assess that, we evaluated combinations of parameters by comparing them with the lexicon. We finally came to the conclusion that, for this task, the most useful parameter combination consisted in assigning a very high weight to structural and discourse-contextual information, and a relatively important weight to content and lengh, while no weight at all was assigned to frequency or mutual information. This combination of parameters also provides an empirical approach to the delimitation of the concept of DM, by eliciting the most influential among a set of DM-characterising features.

However, the evaluation of parameters failed to capture the number of DMs non present in the lexicon retrieved by each parameter or combination of parameters. To do that, the highest ranked DM candidates of each of the lists obtained for each parameter or parameter combination should have been revised manually. That's why only the best combinations of parameters were evaluated as to the enhancement of the lexicon they provided.

3.3. Results with combined parameters

In Table 2 the results of the evaluation of X-TRACTOR and the mentioned baseline are presented. From the sample of 350,000 words, the baseline obtained a list of 60,155 DM candidates, while X-TRACTOR proposed 269,824. Obviously, not all of these were actual DMs, but both systems present an 88% coverage of the DMs in the lexicon that are present in this corpus sample, which were 372.

Concerning goodness of DM assessment, it can be seen that 43% of the 100 DM candidates ranked highest by the baseline were or contained actual DMs, while X-TRACTOR achieved a 68%. Out of these, the baseline succeeded in identifying a 9% of DMs that were not in the lexicon, while X-TRACTOR identified a 15%. Moreover, X-TRACTOR identified an 8% of temporal expressions. The fact that they are identified by the same features characterising DMs indicates that they are very likely to be treated in the same way, in spite of heterogeneous discursive content.

In general terms, it can be said that, for this task, X-TRACTOR outperformed the baseline, suceeded in enlarging an initial DM lexicon and obtained quality results and low noise. It seems clear, however, that the dummy baseline is useful for locating DMs in text, although it provides a limited number of them.

4. Conclusions and Future Directions

By this application of X-TRACTOR to a DM extraction task for Spanish, we have shown that bootstrap-based lexical acquisition is a valid method for enhancing a lexicon of DMs, thus improving the limited coverage of the starting resource. The resulting lexicon exploits the properties of the input corpus, so it is highly portable to restricted domains. This high portability can be understood as an equivalent of domain independence.

The use of this empirical methodology circumvents the bias of human judges, and elicits the contribution of a number of parameters to the identification of DMs. Therefore, it can be considered as a data-driven delimitation of the concept of DM. However, the impact of the enhancement obtained by bootstraping the lexicon should be assessed in terms of prototypicality, that is to say, it should be studied how enlarging a starting set of clearly protoypical DMs may lead to finding less and less prototypical DMs. For an approach to DM prototypicality, see (Alonso et al., 2002).

Future improvements of this tool include applying techinques for interpolation of variables, so that the tuning of the parameters for *markerhood* assessment can be carried out automatically. Also the process of rule induction from the lexicon to the rule module can be automatised, given classes of DM-constituting-words and classes of DMs. Moreover, it has to be evaluated in bigger corpora.

Another line of work consists in exploiting other kinds of knowledge for DM extraction and ponderation. For example, annotated corpora could be used as input, tagged with morphological, syntactical, semantic or even discursive information. The resulting DM candidate list could be pruned by removing proper nouns from it, for example, with the aid of a proper noun data base or *gazetteer* (Arévalo et al., 2002).

To test the portability of the system, it should be applied to other tasks and languages. An experiment to build a DM lexicon for Catalan is currently under progress. To do that, we will try to alternative strategies: one, translating the linguistic knowledge modules to Catalan and directly applying X-TRACTOR to a Catalan corpus, and another, obtaining an initial lexicon by applying the dummy baseline presented here and carrying out the whole bootstrap process.

5. Acknowledgements

This research has been conducted thanks to a grant associated to the X-TRACT project, PB98-1226 of the Spanish Research Department. It has also been partially funded by projects HERMES (TIC2000-0335-C03-02) and PETRA (TIC2000-1735-C02-02).

6. References

Laura Alonso, Irene Castellón, Lluís Padró, and Karina Gibert. 2002. Clustering discourse markers. submitted.

Montse Arévalo, Xavi Carreras, Lluís Màrquez, M.Antònia Martí, Lluís Padró, and M.José Simón. 2002. A proposal for wide-coverage spanish named entity recognition. Technical Report LSI-02-30-R, Dept. LSI, Universitat Politècnica de Catalunya, Barcelona, Spain.

Robert Dale and Alistair Knott. 1995. Using linguistic phenomena to motivate a set of coherence relations. *Discourse Processes*, 18(1):35–62.

C. Engehard and L. Pantera. 1994. Automatic natural acquisition of a terminology. *Journal of Quantitative Linguistics*, 2(1):27–32.

Kyo Kageura and Bin Umino. 1996. Methods of automatic term recognition: A review. *Terminolgy*, 3(2):259–289.

Jung Hee Kim, Michael Glass, and Martha W. Evens. 2000. Learning use of discourse markers in tutorial dialogue for an intelligent tutoring system. In *COGSCI 2000, Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, Philadelphia, PA.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organisation. *Text*, 3(8):234–281.

Daniel Marcu. 1997. From discourse structures to text summaries. In Mani and Maybury, editors, *Advances in Automatic Text Summarization*, pages 82 – 88.

Jorge Vivaldi. 2001. *Extracción de candidatos a término mediante combinación de estrategias heterog éneas*. Ph.D. thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.