A Statistical Approach towards Unknown Word Type Prediction for Deep Grammars

Yi Zhang and **Valia Kordoni**
Department of Computational Linguistics
Saarland University
Saarbrücken, Germany, D-66041

Abstract

This paper presents a statistical approach to unknown word type prediction for a deep HPSG grammar. Our motivation is to enhance robustness in deep processing. With a predictor which predicts lexical types for unknown words according to the context, new lexical entries can be generated on the fly. The predictor is a maximum entropy based classifier trained on a HPSG treebank. By exploring various feature templates and the feedback from parse disambiguation results, the predictor achieves precision over 60%. The models are general enough to be applied to other constraint-based grammar formalisms.

1 Introduction

Deep processing delivers fine-grained syntactic and semantic analyses which are desirable for advanced NLP applications. However, *specificity* and *robustness* are the major difficulties that deep processing has encountered for years.

Unlike shallow methods, which in most cases deliver an expected number of analyses, the number of output from deep processing is usually unpredictable, especially for open texts. The *specificity* problem arises when there are more analyses generated than expected. The analyses might be linguistically sound, but practically uninteresting for real applications. Recently, with more deep processing resources made available (Oepen et al., 2002), the specificity problem is being alleviated with statistical parse selection models (Toutanova et al., 2002).

As to *robustness*, more open questions remain to be investigated. A deep grammar is normally a complicated rule system. Whenever the input varies, even slightly, beyond the grammar developers' expectations, the output becomes unpredictable.

Closer studies of deep grammars have shown that lexicon coverage is one of the major barriers preventing deep grammars from being used for open text processing. Take the LinGO English Resource Grammar (ERG) (Copestake and Flickinger, 2000), for instance. The grammar has been developed for more than 10 years, and currently contains about 22K lexicon entries. A recent test on the BNC corpus reported that only 32% of the strings have full lexical span, of which 57% get at least one parse (Baldwin et al., 2004). About 40% of the parsing failures are due to lexicon missing. Lexicalized deep grammars rely on knowledge-rich lexicon. However, the construction of a lexicon with decent coverage requires a huge amount of human effort and considerable linguistic proficiency.

A widely adopted approach towards robust deep processing is to integrate shallow methods (Callmeier et al., 2004). However, most recent approaches still work on various fall-back strategies. When a deep processing component fails to deliver output, intermediate or shallow components are invoked to provide compatible analyses. Practically valid, this approach does not directly help to enhance the robustness of deep processing itself.

Inspired by the statistical approaches in parse selection (Toutanova et al., 2002), we propose a statistical approach for unknown word type prediction. The experiments are carried out on a broad-coverage linguistically-precise HPSG grammar for English, the LinGO English Resource Grammar (ERG) (Copestake and Flickinger, 2000). However the underlying statistical model is general enough to apply to other deep grammars. Also, by incorporating the parse disambiguation result, we show that the *robustness* is in nature a dual problem to the *specificity*. And they can benefit from each other's improvements.

The remainder of the paper is structured as follows: Section 2 gives the background about the lexicon in HPSG; Section 3 describes our statistical models for unknown word type prediction and the various feature templates we

Proceedings of the Australasian Language Technology Workshop 2005, pages 24–31,
Sydney, Australia, December 2005.

use; Section 4 shows how the parse selection model can be incorporated to enhance the precision of prediction ; Section 5 reports on the experiment results; Section 6 compares our approach to other related work; Section 7 concludes our approach and presents some aspects of our future work.

2 Lexicon Representation and Definitions in HPSG

Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) is a widely adopted constraint-based grammar formalism. Based on *typed feature structure (TFS)* (Carpenter, 1992), HPSG is highly lexicalized, which means there is only a limited number of highly generalized rules (ID Schemata & LP rules). A knowledge-rich lexicon is organized into a complex type hierarchy.

In HPSG, all the linguistic objects are modeled by TFSs. Formally, a TFS is a *directed acyclic graph (DAG)*. Each node in the DAG is labelled with a *sort symbol* (or *type*) corresponding to the category of the linguistic object. All the *sort symbols* are organized into an inheritance system, namely the *type hierarchy*. Two types are compatible if they share at least one common subtype in the hierarchy.

The lexicon is also organized into the type hierarchy. In principle, each lexical entry is a well-formed TFS, which conveys a set of constraints. The constraints include both feature-value appropriateness and type compatibility. For instance, Figure 1 is the TFS for the proper name *"Mary"*.

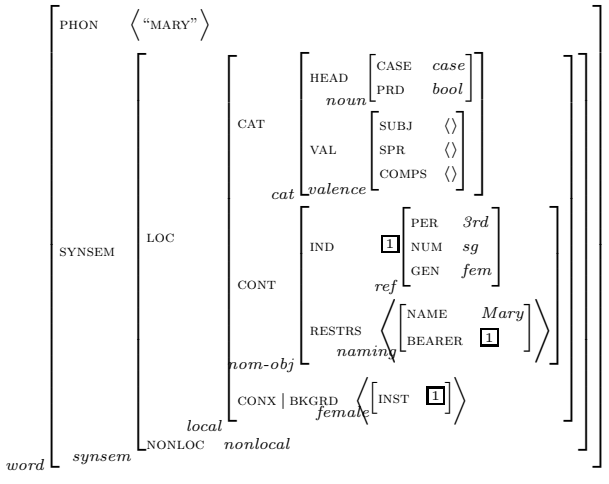

Figure 1: TFS of lexical entry for *"Mary"*

However in implementation, complete description is barely necessary. Most of the constraints will be conveyed via type inheritance. Only entry specific information like the stem and the semantic relation are required.

```
tsunami_n1 := n_intr_le &
  [ STEM < "tsunami" >,
    SYNSEM [ LKEYS.KEYREL.PRED
               "_tsunami_n_rel",
             PHON.ONSET con ] ].

admire_v1 := v_np_trans_le &
 [ STEM < "admire" >,
   SYNSEM [ LKEYS.KEYREL.PRED
              "_admire_v_rel",
            PHON.ONSET voc ] ].
```

Figure 2 gives part of the lexical hierarchy in ERG under the type *basic_noun_word*. Types with the suffix *"le"* are so-called leaf lexical types and should be directly assigned to lexical entries. These types are always mutually separated and incompatible. It is noticeable that each lexical entry takes exactly one leaf lexical type. When a word has more than one syntactic and/or semantic behaviors, different lexical entries will be created separately.

ERG[1] defines in total 741 leaf lexical types, of which 709 types are actually used in its lexicon with 12347 entries. A large number of these lexical types are closed categories whose lexical entries should already exist in the grammar. It is obvious that missing lexical entries, in most cases, should be in open categories. Verb, noun, adjective and adverb are the major open categories. In ERG, the number of leaf lexical types under these general categories are shown in Table 1.

General Cat.	Leaf Lex Types Num.
verb	261
noun	177
adjective	78
adverb	53

Table 1: Number of Leaf Lexical Types under Major Open Categories in ERG

However, even for the open categories, the distribution of existing lexical entries over different lexical types varies significantly. Table

[1]The June 2004 release of ERG was used throughout this paper for experiments and statistics. This was also the version used for building the latest version of Redwood Treebank.

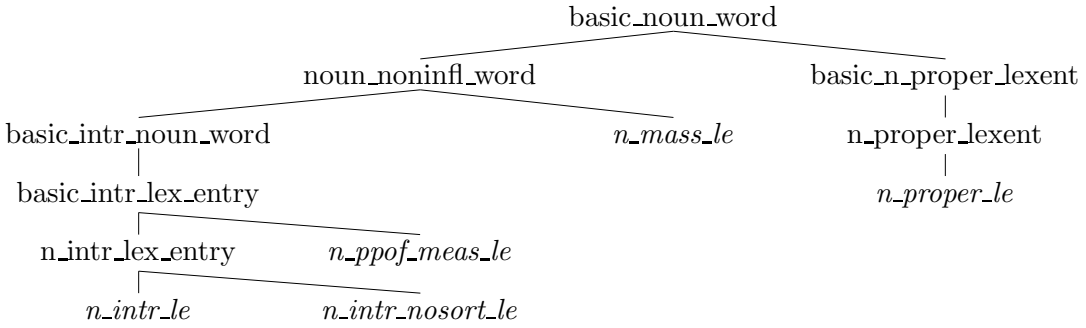

Figure 2: Part of the Lexical Hierarchy in ERG

2 lists the top 10 lexical types with maximum number of entries in the ERG lexicon.

Leaf Lexical Type	Num. of Entries
n_intr_le	1742
n_proper_le	1463
adj_intrans_le	1386
v_np_trans_le	732
n_ppof_le	728
adv_int_vp_le	390
*v_np*_trans_le*	342
n_mass_count_le	292
v_particle_np_le	242
n_mass_le	226

Table 2: Number of Entries for Top-10 Leaf Lexical Types in ERG

The top 10 verbal types count for about 75% of the verbal entries. For nouns the figure is about 95% and 90% for adjectives. Presumably, the automated lexical extension for nouns will be easier. This is plausible because verbal lexical entries normally require more detailed subcategorization information.

3 Statistical Unknown Word Type Prediction Models

For open text processing, a static lexicon inevitably becomes insufficient. A better strategy is to build an unknown word type predictor which can *"guess"* the lexical type from the available context, and generate lexical entries on the fly.

As mentioned in Section 2, the lexicon of an HPSG grammar is organized into a type hierarchy. Each entry bears exactly one leaf lexical type. So the predictor is actually a classifier, which takes various context and morphological forms of the unknown word into consideration, and picks out the most suitable leaf lexical type as output.

Such an unknown word type predictor is essentially very similar to a part-of-speech (POS) tagger. A typical POS tagger assigns a (unique or ambiguous) part-of-speech tag to each token in the input. A large number of current language processing systems use a POS tagger for pre-processing. The difference is that our unknown word type predictor has a very larger tagset. The tagset of a typical POS tagger usually contains tens of different tags. But our predictor needs to handle hundreds of possible types. In addition, an unknown word type predictor only predicts unknown words while a typical POS tagger generates tags for each token on the input sequence. Another point is that our unknown word type predictor can use any context information available at the processing stage. But normally a POS tagger only uses surface context features because these are usually used during pre-processing.

3.1 Maximum Entropy Classifier Based Predition Model

Considering these difference, we have constructed our predictor based on a maximum entropy classifier. The advantages of a Maximum entropy model lie in the general feature representation and in no independence assumptions between features. A maximum entropy model can also easily handle thousands of features and large numbers of possible outputs.

For our prediction model, the probability of a lexical type t given an unknown word and its context c is:

$$p(t,c) = \frac{exp(\sum_i \theta_i f_i(t,c))}{\sum_{t' \in T} exp(\sum_i \theta_i f_i(t',c))} \qquad (1)$$

where feature $f_i(t,c)$ may encode arbitrary

characteristics of the context. The parameters $< \theta_1, \theta_2, \ldots >$ can be evaluated by maximizing the pseudo-likelihood on a training corpus (see (Malouf, 2002)).

The basic feature templates used in our ME-based model include the prefix and suffix of the unknown word, the context words within a window size of 5, and their corresponding lexical types.

3.2 Using Partial Parsing Results as Features

Each lexical type is essentially a set of constraints on linguistic objects. If a word has a specific lexical type, it must conform to all the constraints demanded by the type, and hence it can only appear in some specific linguistic context. The constraints concern various linguistic aspects, among which syntactic constraints are predominant.

One advantage of using a maximum entropy based model is that ME allows the combination of diverse forms of contextual information in a principled manner, and it does not impose any distributional assumptions on the training data. So far, only the surface context features (words and their lexical types) are used. It can be presumed that the precision can be enhanced by adding syntactic context as features into the prediction model.

However, syntactic information is not available in a traditional pipeline processing model, where the syntactic analysis will be the post-processing module to the predictor. Also, when there are unknown words in the input, a full analysis of the sentence is not possible.

So we have modified our strategy by inserting a partial parsing stage before the lexical type predictor if there are unknown words on the input sequence.

The partial parse needs some clarification. A full parse can be represented by a set of edges as shown in Figure 3(a). Each edge is derived from a rule application. There is no more than one edge between each pair of positions. And there is always exactly one full span edge in a full parse.

A partial parse of an input sequence is a set of edges which composes a shortest path from the beginning to the end of the sequence[2]. There might be more than one partial parse for a given input sequence. As shown in Figure 3(b), when the word between position 2 and 3 is unknown, a dummy edge c is created. This dummy edge will prevent further rule application. Both $a - c - d$ and $b - c - d$ are partial parses.

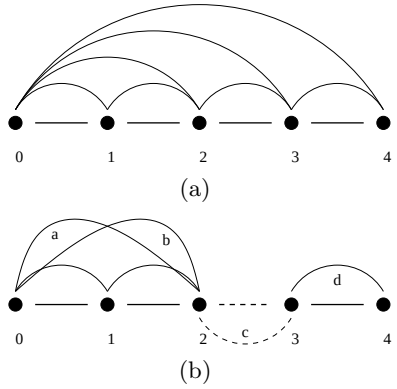

Figure 3: Parsing edges: (a) edges in a full parse; (b) edges in partial parses.

From the partial parses, we collect all edges that are adjacent to the left/right of the unknown word, respectively. Then the rules that generate these edges are counted according to their application (once per edge). The most frequently used rules to create left/right adjacent edges are added as two features conveying syntactic information into the ME-based model. A complete list of all features templates used in our predictor are listed in Table 3.

4 Incorporating Parse Disambiguation Results

As mentioned before, deep lexical types normally encode complicated constraints that only make sense when they work together with the grammar rules. And some subtle differences between lexical types do not show statistical significance in a corpus with limited size. So the feedback from later stages of deep processing is very important for predicting the lexical types for the unknown words.

The partial parsing results break the pipeline model. However, they might help only when the unknown is not the head of the phrase. Otherwise, the full parse crushes into small fragments, and the partial parsing results normally make no sense. An alternative way of breaking the pipeline model is to help the parser to generate full parses in the first place, and let the parsing result tell which lexical entry is good.

[2]Note that the edges on the full parse of the sentence are not necessary in the corresponding partial parses if a word is assumed to be unknown. However, partial parses do reduce the number of candidate edges for consideration.

Features
X is prefix of w_i, $\|X\| \leq 4$
X is suffix of w_i, $\|X\| \leq 4$
$t_{i-1} = X$, $t_{i-2}t_{i-1} = XY$, $t_{i+1} = X$, $t_{i+1}t_{i+2} = XY$
$w_{i-2} = X$, $w_{i-1} = X$, $w_{i+1} = X$, $w_{i+2} = X$
LP is the left adjacent most frequent edge of w_i
RP is the right adjacent most frequent edge of w_i

Table 3: Feature templates used in ME-based prediction model for word w_i (t_j is the lexical type of w_j)

In order to help the parser to generate a full parse of the sentence, we feed the newly generated lexical entries directly into the parser. Instead of generating only one entry for each occurrence of unknown, we pass on top n most likely lexical entries. With these new entries, the sentence will receive one or more parses (assuming the sentence is grammatical and covered by the grammar). From the parsing results, a best parse is selected with the disambiguation model, and the corresponding lexical entry is taken as the final result of lexical extension.

Within this processing model, the incorrect types will be ruled out if they are not compatible with the syntactic context. Also the infrequent readings of the unknown will be dispreferred by the disambiguation model.

5 Experiments

Missing lexical entries can be discovered by lexicon checking. Precision is the only measurement for the lexical type predictor. In this section we will evaluate our models by experiments.

5.1 Resources

Redwoods (Oepen et al., 2002) is a HPSG treebank that records full analyses of sentences with *ERG*. The genre of texts includes email correspondence, travel planning dialogs, etc. The 5th growth of *Redwoods* contains about 16.5K sentences and 122K tokens[3].

In all our experiments, we have done a 10-fold cross validation on the *Redwoods* treebank. For each fold, words that do not occur in the training partition are assumed to be unknown.

A modified version of the efficient HPSG parser *PET* (Callmeier, 2000; Callmeier, 2001) has been used to generate the derivation tree fragments of the partial parses.

We have also modified *LexDB* (Copestake et al., 2004) in order to be able to add temporal lexical entries that are only active for specific sentence.

The parse disambiguation model we have used is a maximum entropy based model that uses non-lexicalized features with 2 levels of grandparnets (see (Toutanova et al., 2002) for detailed discussion about parse disambiguation models for HPSG grammars).

For maximum entropy parameter estimation, we have used (Malouf, 2002)'s *MaxEnt* package.

5.2 Results

For comparison, we have built a baseline system that always assigns a majority type to each unknown according to the POS tag. More specificically, we tag the input sentence with a small Penn Treebank-like POS tagset. Then POS tag is mapped to a most popular lexical type for that POS.[4] Table 4 lists part of the mappings.

POS	Majority Lexical Type
noun	n_intr_le
verb	v_np_trans_le
adj.	adj_intrans_le
adv.	adv_int_vp_le

Table 4: Part of the POS tags to lexical types mapping

Again for comparison, we have built another two simple prediction models with two popular general-purpose POS taggers, *TnT* and *MXPOST*. *TnT* is a HMM-based trigram tagger while *MXPOST* is maximum entropy based. We have trained the tagging models by using all the leaf lexical types as the tagset. The taggers tag the whole sentence. But only the output tags for the unknowns are taken to generate the lexical entries.

[3]Sentences without a full analysis are neither counted here nor used in experiments.

[4]This is similar to the built-in unknown word handling mechanism of the *PET* system.

The maximum entropy based model is tested both with and without using partial parsing results as features. To incorporate disambiguation results, our predictor generates 3 entries for each unknown and store them as temporary entries into the *LexDB*.

Precisions of the different prediction models are shown in Table 5.

Model	Precision
Baseline	30.7%
TnT	40.4%
MXPOST	40.2%
ME(-pp)	50.0%
ME(+pp)	50.5%
ME(-pp)+ disambi. result	61.3%

Table 5: Precision of Unknown Word Type Predictors (+/-pp means w or w/o partial parsing result features)

The baseline model achieves precision around 30%. This means that the task of unknown word type prediction for deep grammars is non-trivial. The general-purpose POS taggers based models perform quite well, outperforming the baseline by 10%. As a confirmation to (Elworthy, 1995)'s claim, a huge tagset does not imply that tagging will be very difficult. Our ME-based model significantly outperforms the taggers-based models by another 10%. This is a strong indication of our model's advantages.

By incorporating simple syntactic information into the ME-based model, we get extra precision gain of less than 1%. It is worth noticing that the syntactic features we used are still naive. Better syntactic features remain to be explored in future work. Also, by applying partial parsing, the computation complexity increases significantly in comparison to our basic ME-based model.

By incorporating the disambiguation results, the precision of the model boosts up for another 10%. The computational overhead is proportional to the number of candidate entries added for each unknown word. However, in most cases, introducing lexical entries with incorrect types will end up to parsing failure and can be efficiently detected by quick checking. In such cases the slowdown is acceptable.

In general, we have achieved up to 60% precision of unknown word type prediction for the ERG in these experiments. Given the complexity of the grammar and the huge number of possible lexical types, these results are satisfying. Also, in real case of grammar adaptation for new domains, a large portion of unknowns are proper names. This means that the precision might get even higher in real applications. A test with some small text collection with real unknown words [5] shows that the precision can easily go above 80% with the basic ME model without partial parsing features.

It should also be mentioned that some of these experiments are also carried out for Dutch Alpino Grammar (Bouma et al., 2001), and similar results are obtained. This shows that our method may be grammar and platform independent.

6 Comparison with Related Work

This work is in essence very similar to the work of deep lexical acquisition (DLA) in (Baldwin, 2005). A minor difference is that our model always generates (at least) one lexical entry for the unknown, so that the deep processing does not halt at the very beginning. A more important difference is that, while (Baldwin, 2005) focuses on generalizing the method of deriving DLA models on various secondary language resources, our work focuses more on how to utilize the deep grammar itself as a source for enhancing robustness. The *Redwoods Treebank* is by nature the output of the deep grammar. And the parsing, as well as the disambiguation models are also part of the grammar that has eventually contributed to the unknown word type prediction.

(Erbach, 1990; Barg and Walther, 1998; Fouvry, 2003) followed a different approach towards unknown words processing for unification based grammars. The basic idea was to use the underspecified lexical entries, namely TFSs with fewer constraints, in order to generate full parses for the sentences, and then extract the sub-TFS from the parses as a new lexical entry. However, lexical entries generated in this way might be both too general and too specific. And underspecified lexical entries with fewer constraints allow more grammar rules to be applied while parsing. It gets even worse when

[5] We used a text set named *rondane* for training and *hike* for testing. *rondane* contains 1424 sentences in formal written English about tourism in the norwegian mountain area, with an average sentence length of 16 words; *hike* contains 320 sentences about outdoor hiking in Norway with an average sentence length of 14.3 words. Both contain a lot of unknowns like location names, transliterations, etc.

two unknown words occur next to each other, which might allow almost any constituent to be constructed. Also, the underspecified lexical entry significantly increases computational complexity. (van Schagen and Knott, 2004) took a similar approach of interactive unknown word acquisition in a dialogue context.

(Thede and Harper, 1997) reported an empirical approach towards unknown lexical analysis using morphological and syntactic information. The approach is similar to ours in spirit. However, the experiments were done for a shallow parser with a very limited number of word classes. The applicability to lexicalist deep grammars with lots of lexical types is unknown.

In (Malouf and van Noord, 2004), the maximum entropy models were used for wide coverage parsing with the *Alpino* Dutch grammar (Bouma et al., 2001). But the focus was on parse selection, not unknown words processing.

Another related work is *supertagging* (Bangalore and Joshi, 1999). In supertagging, the lexical items are assigned with rich descriptions (supertags) that impose complex constraints in a local context. Some statistical techniques of assigning supertags to unknown words have been reported. For example, (Bangalore and Joshi, 1999) used a simple method of combining a probability estimate for unknown words $P(UKN|T_i)$ with a probability estimate based on word features (capitalization, hyphenation, ending of words) by:

$$P(W_i|T_i) = P(UNK|T_i) * P(w\_feat(W_i)|T_i) \quad (2)$$

where UNK is a token associated with each supertag and its count N_{UNK} is estimated by:

$$P(UNK|T_j) = \frac{N_1(T_j)}{N(T_j)+\eta} \quad (3)$$

$$N_{unk}(T_j) = \frac{P(UNK|T_j) * N(T_j)}{1 - P(UNK|T_j)} \quad (4)$$

$N_1(T_j)$ is the number of words that are associated with the supertag T_j that appear in the corpus once. From some aspect, this approach is similar to our work. But our ME-based model allows more general feature representation. Also the lexical types we used are more general in the sense that both local and non-local constraints are encoded.

7 Conclusion and Future Work

Several statistical unknown word type prediction models are implemented and evaluated for deep HPSG grammars. The general-purpose POS taggers based approach delivers satisfying precision. The maximum entropy based predictor allows for more general feature representation. By incorporating parse disambiguation results, the unknown word type predictor achieves precision over 60%.

Although the experiments are carried out with the ERG, the underlying model is general enough to be easily applied on other constraint-based lexicalist grammars, provided the lexical categories can be abstracted by a set of atomic types.

Several aspects of this work need further exploration. More sophisticated syntactic features should be investigated. Besides, the deep grammar also provides semantic analyses which are not available in shallow processing. The general feature representation in our model allows the incorporation of this orthogonal dimension of information to enhance the precision of prediction. Also, larger corpora in more variety of genres are certain to generate better models. The application of the method to more deep grammars is anticipated.

References

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Timothy Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Petra Barg and Markus Walther. 1998. Processing unkonwn words in HPSG. In *Proceedings of the 36th Conference of the ACL and the 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of dutch. In *Computational Linguistics in The Netherlands 2000*.

Ulrich Callmeier, Andreas Eisele, Ulrich

Schäfer, and Melanie Siegel. 2004. The deepthought core architecture framework. In *Proceedings of LREC 04*, Lisbon, Portugal.

Ulrich Callmeier. 2000. PET – a platform for experimentation with efficient HPSG processing techniques. *Journal of Natural Language Engineering*, 6(1):99–108.

Ulrich Callmeier. 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge University Press, Cambridge, England.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage english grammar using hpsg. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Ann Copestake, Fabre Lambeau, Benjamin Waldron, Francis Bond, Dan Flickinger, and Stephan Oepen. 2004. A lexicon module for a grammar development environment. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal.

David Elworthy. 1995. Tagset design and inflected languages. In *EACL SIGDAT workshop "From Texts to Tags: Issues in Multilingual Language Analysis"*, pages 1–10, Dublin, Ireland, April.

Gregor Erbach. 1990. Syntactic processing of unknown words. IWBS Report 131, IBM, Stuttgart.

Frederik Fouvry. 2003. Lexicon acquisition with a large-coverage unification-based grammar. In *Companion to the 10th of EACL*, pages 87–90, ACL, Budapest, Hungary.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses*.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conferencde on Natural Language Learning (CoNLL-2002)*, pages 49–55.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes*, Taipei.

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

Scott M. Thede and Mary Harper. 1997. Analysis of unknown lexical items using morphological and syntactic information with the timit corpus. In *Proceedings of the Fifth Workshop on Very Large Corpora*, pages 261–272.

Kristina Toutanova, Christoper D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse ranking for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT2002)*, pages 253–263, Sozopol, Bulgaria.

Maarten van Schagen and Alistair Knott. 2004. Tauira: A tool for acquiring unknown words in a dialogue context. In *Proceedings of the 2004 Australasian Language Technology Workshop (ALTW2004)*, Macquarie University, Australia.