An Empirical Comparison of Unknown Word Prediction Methods

Kostadin Cholakov[†], **Gertjan van Noord**[†], **Valia Kordoni**[‡], **Yi Zhang**[‡]

† University of Groningen, The Netherlands
‡ Saarland University and DFKI GmbH, Germany
{k.cholakov,g.j.m.van.noord}@rug.nl
{kordoni,yzhang}@coli.uni-sb.de

Abstract

We compare two types of methods which deal with unknown words in the context of computational grammars. Methods of the first type are based on the idea of *supertagging* and use a tagger to predict lexical descriptions for unknown tokens in a given input. The second type of methods perform *lexical acquisition* (LA) which, in the context of this paper, refers to the automatic acquisition of new lexical entries for the lexicon of a given grammar. The methods are compared based on the effect their application has on the parsing coverage and accuracy of the GG grammar of German (Crysmann, 2003). In particular, we adapt the LA method of Cholakov and van Noord (2010) which was originally developed for the Dutch Alpino system to be used with the GG. Its impact on coverage and accuracy on a test corpus of German newspaper texts is compared to the results reported previously on the same corpus for methods which employed a tagger. Furthermore, in a smaller experiment, we show that the linguistic knowledge this LA method provides can also be used for sentence realisation.

1 Introduction

Computational grammars of natural language which rely on hand-crafted lexicons containing elaborate linguistic descriptions usually have low lexical coverage. This is due to the fact that it is impossible to list every word in a language in the lexicon. If a word is unknown to the grammar, or its description in the lexicon is wrong or incomplete, the grammar might fail to produce a (correct) analysis.

In this paper, we compare two types of approaches for dealing with unknown words. The first type is based on the concept of supertagging while the second one performs LA. Generally, supertagging refers to the process of applying a sequential tagger to assign lexical descriptions associated with each word in an input string, relative to a given grammar. It was introduced as a means to reduce parsing ambiguity of LTAG grammars (Bangalore and Joshi, 1999), and has since been applied within CCG (Clark, 2002; Clark and Curran, 2004) and HPSG (Dridan et al., 2008; Zhang et al., 2010) grammars. Supertagging has also been employed for dealing with unknown words. However, in such methods, the tagger is used to assign lexical descriptions **only** to the unknown tokens in a given sentence. It is important to note here that henceforth, we will use the term *suppertagging* in this narrow sense of tagging unknown words only. Supertagging methods often work *online*. The unknown words are assigned lexical entries when they are encountered in the input during parsing. Therefore, the focus is primarily on improving the parsing coverage and accuracy of the grammar for a particular input. The performance of such methods is usually evaluated in terms of *token accuracy*, that is the proportion of correctly tagged unknown lexical tokens in the input.

LA, on the other hand, aims at learning automatically new lexical entries and thus, at extending the lexicon of a given grammar. The strategy is to improve the parsing coverage and accuracy by improving the quality and the coverage of the lexicon of the grammar. While suppertagging methods are concerned with a particular form of the unknown word within the particular context this form occurs in the input, LA techniques look at various forms and contexts of the unknown word and use more detailed linguistic information to learn a new lexical entry for it. This makes LA methods more complex and time-consuming. That is why, they are often applied *offline*. LA methods are usually

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 767–775,
Chiang Mai, Thailand, November 8 – 13, 2011.

evaluated in terms of *type precision* and *type recall*. For a given LA method, type precision indicates the proportion of correctly predicted lexical entries and type recall indicates how many of the correct lexical entries for a given word are actually found.

Both supertagging and LA have been successfully used. For instance, Blunsom and Baldwin (2006) employ a conditional random field-based tagger to predict lexical entries for large-scale HPSG grammars of English (ERG; (Copestake and Flickinger, 2000)) and Japanese (JACY; (Siegel and Bender, 2002)). Zhang and Kordoni (2006) and Dridan et al. (2008) have developed a maximum entropy-based (ME) tagger and investigate the effect its application has on the parsing performance of the ERG and the GG. Other methods which apply the same tagger to the GG include Nicholson et al. (2008) and Cholakov et al. (2008). The ERG, the GG and JACY are part of the DELPH-IN collaboration[1] and as such, they share the same grammar design and parsing architecture which facilitates the application of the same tagger. Baldwin (2005) presents a LA approach where various secondary resources (POS taggers, chunkers, etc.) are used to create an abstraction of words unknown to the ERG and then binary classifiers are employed to learn lexical entries for those words. However, learning is done based on incomplete information obtained by the various resources used. Further, no evaluation of the effect the method has on the parsing performance of the ERG is provided. Cholakov and van Noord (2010) describe a LA method where a ME-based classifier is used to acquire lexical entries for all forms in the paradigms of words, unknown to the large-scale Dutch Alpino grammar (van Noord, 2006). The paper also shows that the application of LA improves the parsing accuracy of Alpino on open-domain texts.

For our purpose of comparing supertagging and LA, we investigate how methods implementing these strategies affect the parsing coverage and accuracy when applied with the same grammar, namely the GG. Our choice of German and the GG as a platform for comparison is motivated by the fact that German has richer morphology and a more free word order than both English and Dutch which makes unknown word handling more challenging. We adapt the LA method of Cholakov and van Noord (2010) – henceforth C&vN– and applied with the GG. Then, we compare the parsing coverage and accuracy the GG achieves on a German newspaper corpus to those reported for supertagging methods applied previously with the GG on the same corpus. For all techniques, the experiments show that their application leads to an increase in coverage compared to the baseline, that is the standard GG setup. However, the supertagging methods achieved this at the price of having lower accuracy than the baseline. The application of the adapted C&vN method, on the other hand, increases parsing accuracy compared to the baseline. This difference in quality might not always be crucial since less accurate parses produced by the grammar can still be used successfully in many NLP applications. In such cases, the less complex supertagging methods might be the preferred choice. However, through a small sentence realisation experiment, we give an example of an application where high-quality LA is a prerequisite.

Other kinds of LA techniques have also been proposed. Cussens and Pulman (2000) used a symbolic approach employing *inductive logic programming*, while Erbach (1990), Barg and Walther (1998) and Fouvry (2003) followed a unification-based approach. However, it is doubtful if those methods are scalable since they have not been applied to large-scale grammars and no meaningful evaluation has been provided.

The remainder of the paper is organised as follows. Section 2 describes the resources we employ. Section 3 gives an overview of the supertagging methods previously applied with the GG. Section 4 describes the adaptation of the C&vN method to the GG. Section 5 gives details on the training procedure for the ME-based classifier used in the C&vN technique. Section 6 evaluates the parsing coverage and accuracy on the German newspaper corpus when this technique is applied with the GG and compares the results to the results reported previously for the suppertagging methods for this corpus. Section 7 explores the possibility of using newly acquired lexical entries in a small sentence realisation task. Section 8 concludes the paper.

2 Resources

The GG (Crysmann, 2003) is an HPSG grammar based on typed feature structures. The GG types are strictly defined within a type hierarchy. The

[1] http://wiki.delph-in.net/

grammar contains constructional and lexical rules, as well as a lexicon where words are assigned lexical types. Currently, it consists of 5K types, 115 rules and the lexicon contains approximately 55K entries. There are 411 distinct lexical types which words can be mapped onto. Below is an example of a GG lexical entry, namely the entry for the word *Aufgabe* (a task):

```
aufgabe-n := count-noun-le &
 [ MORPH.LIST.FIRST.STEM < "Aufgabe" >,
   SYNSEM.LKEYS [ --SUBJOPT -,
            KEYAGR c-n-f,
            KEYREL "_aufgabe_n_rel",
            KEYSORT abstract,
            MCLASS nclass-9 ] ].
```

The lexical type 'count-noun-le' shows that the word is a countable noun[2]. The STEM type feature specifies the stem of the word. The stem for nouns is the singular nominative noun form, for adjectives– the base noninflected form and for verbs– the root form. Adverbs in German have a single form which is used as the value of the STEM feature in adverb entries. Some nouns (e.g., *Baukosten* (building costs)) do not have all forms typical for German nouns. In such cases, the word itself is set as the value of the STEM feature. The KEYAGR feature indicates case, number and gender. In the example above, case and number are left underspecified while the gender is set to feminine. The value of SUBJOPT shows that this noun is always used with an article and MCLASS indicates its morphological paradigm. The KEYREL and KEYSORT features define the semantics of the word.

Further, we employ the PET system (Callmeier, 2000) to parse with the GG. PET is a system for efficient processing of unification-based grammars. The system comprises a sophisticated preprocessor, a bottom-up chart parser and a grammar compiler. For our purposes, we should note that the parser has a built-in unknown word guesser which, based on some simple heuristics, assigns generic types to the unknown words. Most of the features in these types are left underspecified.

We compare the results of the various methods based on how they affect the parsing coverage and accuracy of the GG on the Frankfurter Rundschau (FR) newspaper corpus. The corpus contains 614K sentences and the articles in it deal with various domains.

[2] *le* stands for lexeme.

3 Suppertagging Methods

Here, we give an overview of supertagging methods applied previously with the GG. Typically, the unknown words are assigned *open-class* lexical types. In the case of the GG, open-class lexical types are those assigned to nouns, adjectives, verbs and adverbs. It is assumed that closed-class words are already handled by the grammar.

Dridan et al. (2008) adopts the method proposed in Zhang and Kordoni (2006) for English and the ERG and applies it to the GG. A ME-based tagger is trained on features extracted from the context of the unknown word– four characters from the beginning and the end of the word, and two words of context (where available) either side of the target word together with their POS tags[3]. The application of the tagger led to an improved parsing coverage on a test corpus containing 700 short German questions from the CLEF competition. No evaluation in terms of accuracy is provided for German. For the same experiment with the ERG, the accuracy achieved with the tagger was below the baseline accuracy (the standard ERG setup).

Nicholson et al. (2008) employs the same ME-based tagger but the paper examines its performance more closely. The tagger is evaluated by performing a 10-fold cross-validation on the treebank associated with the GG. Words which appeared only in the test fold were considered unknown. Since the sentences in the treebank are annotated, the method is able to use the lexical types of the context words as features instead of their POS tags. The achieved token accuracy is 58%. Then, the tagger is used to predict lexical types for unknown words in a random sample of 1000 sentences extracted from the FR corpus. However, since no annotation is available for the test corpus, the tagger is run with the same features as in Dridan et al. (2008) but without the POS tags of the context words.

The results are given in Table 1. Coverage increased by 8% for the test corpus. An estimation of the parsing accuracy is also given. Parsing accuracy is measured as the proportion of sentences for which a correct parse is produced, among the set of parses. 87 sentences were randomly selected and manually examined. The baseline accuracy was 85%. When suppertagging is applied accuracy drops to 84%. This is consistent with the re-

[3] The corpora used had been POS tagged with the TreeTagger (Schmid, 1994).

sults reported in Dridan et al. (2008) for the ERG.

Cholakov et al. (2008) develops further the idea of using lexical types as features in the tagger. The paper presents the same system setup as Nicholson et al. (2008) but with one crucial difference. The tagset used in the experiments consists of more detailed descriptions of the GG lexical types. In this new tagset, the values of some of the morphosyntactic features from the lexical entries are attached to the type definition, thus making the information they carry accessible to the tagger. For example, according to the guidelines given in Cholakov et al. (2008), the lexical type of *Aufgabe* would become *count-noun-le_-_f* after attaching the values of the SUBJOPT and KEYAGR features to the type definition[4]. Having this additional information, the token accuracy of the tagger increases to 73.54%. The effect the method has on the parsing coverage and accuracy is shown in Table 1. The experiment is done on a random sample of 10K sentences from the FR with the corpus POS tagged and the tags of the context words used as features in the tagger instead of lexical types. Parsing accuracy is measured as the proportion of 100 randomly selected sentences with at a correct parse produced. Again, accuracy decreases in comparison to the baseline (the standard GG setup).

It is important to note that the lexical entries created by the aforementioned methods for unknown words are sort of generic due to the inability of those methods to access the values of the type features defined in the entries. In Nicholson et al. (2008) only the type of the word can be specified and the word is set as the value of the STEM feature. Every other feature in the entry is either underspecified or assigned some default value. The technique of Cholakov et al. (2008) can define more features in the created lexical entry due to the detailed tagset used. However, accuracy still decreased.

4 Lexical Acquisition with The GG

In this section, we describe the adaptation of the C&VN LA method to the GG. This adaptation raises the important question about portability of LA. Because of their complexity and dependence on a particular grammar, LA methods have been applied within a single parsing system, in a single framework, and mostly for a single language. So, it is unclear to what extent the various techniques can be used for a different language or parsing architecture. However, the C&VN method is claimed explicitly to be portable to other systems and languages provided some conditions are met. These include: a finite set of labels which unknown words are mapped onto, a syntactic parser, and a morphological component which generates the paradigm(s) of a given unknown word.

Similar to Cholakov et al. (2008), we created a set of labels which consists of more refined definitions of the GG lexical types. Accordingly, we attach the values of some of the type features defined in the relevant lexical entries to the type definitions of those entries. However, for reasons of data sparseness, we choose to exclude some of the features which Cholakov et al. (2008) considered relevant. Experiments with different sets of features have shown that the best results are achieved when only features designating morphosyntactic agreement are considered. For all noun types and predicative adjectives this is the KEYAGR feature and for verb types allowing for prepositional complements– the COMPAGR and the OCOMPAGR features which indicate the case of the the (oblique) complement. For instance, the lexical type of *Aufgabe* will become *count-noun-le_f* after the gender value of the KEYAGR feature is attached.

C&VN propose a method for the generation of the paradigm(s) of a given unknown Dutch word (Cholakov and van Noord, 2009). The type of word form predicted by the paradigm for that word is used as a feature in the ME-based classifier employed in C&VN for learning unknown words. For example, the paradigm of *Aufgabe* will indicate that the word is a singular feminine noun and this information will be passed on to the classifier. Further, unlike the Dutch Alpino grammar, the GG does not have a full form lexicon. All word forms are derived by applying various morphological rules defined in the GG to the word stem. Therefore, we need to add only the correct stem of the unknown word the lexicon. Then, all forms of the word will automatically be recognised by the grammar. That is why, in the case of the GG, we use the generated paradigms to perform the mapping between unknown words and their stems. For instance, if *Aufgaben* (tasks) is an unknown word, we will add *Aufgabe*, the stem

[4]The values for case and number for nouns are almost always left underspecified; that is why only the value of the gender is attached to the type definition.

Method	Coverage (%)		Accuracy (%)	
	Baseline (standard GG)	GG + tagger	Baseline (standard GG)	GG + tagger
Lexical	12	20	85	84
Lexical+POS	8.9	21.1	85	83

Table 1: Coverage and accuracy results for the GG and the FR corpus. *Lexical* refers to Nicholson et al. (2008) where only affixes and context words are used as features. The last row refers to Cholakov et al. (2008) where the POS tags of the context words are also employed.

indicated by its paradigm, to the lexicon.

Due to the GG design, it is not straightforward to use the morphological rules of the grammar for paradigm generation. Following the C&VN technique developed for generating the paradigms of Dutch words, we created a German finite state morphology. The morphology does not have access to any linguistic information and thus, it generates all possible paradigms allowed by the word orthography. Then, the number of search hits Yahoo! returns for each form in a given paradigm is combined with some simple heuristics to disambiguate the output of the morphology and to determine the correct paradigm(s). We also apply heuristics to guess the gender for words with generated noun paradigms and to determine if a word which is assigned a verb paradigm starts with a separable particle.

One could argue that there is a simpler approach for mapping the various forms of the unknown word to its stem. For instance, the TreeTagger provides both POS and stem information with high accuracy. However, the generation of the paradigms allows us to consider contexts in which other forms of a given unknown word occur and thus, to have access to much more and linguistically diverse data. Further, using the paradigms, we are able to determine which of the morphological classes defined within the GG a given unknown word belongs to and specify the value of the MCLASS type feature in the lexical entry generated for this word. We are also able to determine the value of the VCOMPFORM type feature which indicates the separable particle in verb entries. However, those are not crucial for the classification process and they are likely to cause data sparseness. That is why, those type features are assigned their values after the lexical type of a given word has been predicted. The semantic features in the lexical entries for newly acquired words, as well as the other features whose values are not learnt via LA are assigned default values or left underspecified.

We adopted the ME-based classifier[5] and the features used for unknown word prediction as described in C&VN. The probability of a lexical type *t*, given an unknown word and its context *c* is:

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

where $f_i(t, c)$ may encode arbitrary features from the context and $< \Theta_1, \Theta_2, ... >$ can be evaluated by maximising the pseudo-likelihood on a training corpus (Malouf, 2002).

Table 2 shows the features for *Aufgabe*. Since the stem of the unknown word is added to the lexicon, we also experimented with prefix and suffix features extracted from the stem. We assumed that those could allow for a better generalization of morphological properties but they proved to be less informative for LA.

Features
i) A, Au, Auf, Aufg
ii) e, be, abe, gabe
iii) hyphen_no
iv) noun_feminine #predicted by the paradigm
v) count-noun-le_f, mass-noun-le_f
vi) noun⟨f⟩

Table 2: Features for *Aufgabe* (a task)

Rows **(v)** and **(vi)** show syntactic features obtained from what C&VN refer to as 'parsing with universal types'. In C&VN, each unknown word is assigned all target lexical types, i.e. it is treated as being maximally ambiguous. However, to reduce ambiguity and make the parsing computationally feasible, we use the generated paradigms as a filtering mechanism and assign a given word only those types which belong to the POS of the paradigm(s) generated for this word. That is why, *Aufgabe* is assigned all noun lexical types. Sentences containing morphological forms of *Aufgabe*

[5]The classifier is developed using the TADM tool; `http://tadm.sourceforge.net/`

are then parsed with PET in best-only mode. For each sentence only the best parse selected by the disambiguation model of the parser is preserved. Then, the lexical type that has been assigned to the form of *Aufgabe* occurring in this parse is stored.

We employ the most frequently used type(s), based on an empirical threshold (80% for the GG), as features in the classifier (row **v**). Further, as illustrated in row **(vi)**, each piece of information attached to the type definitions (the part after the underscore) is also taken as a separate feature. By considering these syntactic features, we manage to involve the grammar directly in the prediction process, unlike the methods discussed in the previous section where only lexical features and POS tags were used. Here, the grammar can decide which lexical type(s) are best suited for a given unknown word. This is an effective way to include the syntactic constraints imposed on the unknown word into the prediction process.

5 Training of The Classifier

In order to train the classifier, we need annotated data. That is why, we temporarily remove some words from the GG lexicon and use them for training and tuning the various parameters of the classifier. The lexical entries of the removed words are used as gold standard.

We remove only words belonging to open-class lexical types. Each such type has at least 10 lexical entries in the lexicon mapped onto it and it is assigned to at least 15 distinct words occurring in large corpora parsed with PET and the GG. The parsed corpus we use consists of roughly 2.5M sentences randomly selected from the German part of the Wacky project (Kilgarriff and Grefenstette, 2003). Following these criteria, we have selected 39 open-class types out of the 411 lexical types defined in the GG. The expansion of the type definitions of the 39 types with the values of the relevant type features resulted in the creation of 68 *expanded* types. Table 3 gives more details about the type distribution.

2400 words are removed from the lexicon of the GG. Of these, 2000 are used for training, and 400 words are used as a test set to give an idea about the performance of the classifier. We assume that less frequent words are typically unknown and, in order to simulate their behaviour, all 2400 words have between 40 and 100 occurrences in the parsed corpus. Experiments with a minimum lower than 40 occurrences have shown that this is a reasonable threshold to filter out typos, tokenization errors, etc. The distribution of the parts-of-speech for the 2400 words and the evaluation of the paradigm generation component are shown in Table 4 (some words have more than a single part-of-speech). Accuracy indicates how many of the generated paradigms are correct.

	Original types	Expanded types
Total	39	68
-nouns	5	15
-verbs	28	45
-adjectives	4	6
-adverbs	2	2

Table 3: Distribution of the target lexical types

	overall	nouns	adj	verbs
total	2954	1196	651	694
accuracy(%)	96.45	91.09	100	99.54

Table 4: Paradigm generation results

In the paradigms generated for verbs there were three mistakes. However, the generated verb stems were all correct. Similarly, the stems for all nouns were correct, including the stems of 98 nouns which contained a mistake in their paradigm. In 91 cases the singular genitive form was incorrect, in another 12 cases the predicted gender was wrong. The mapping of the words to their correct stems is correct in all cases.

We allow for multiple types per word to be predicted but we discard the types accounting together for less than 5% of probability mass. Additionally, there are four baseline methods. The ***naive*** one assigns the most frequent expanded type in the lexicon, *count-noun-le_f*, to each unknown word. In the ***naive POS*** baseline each word is given the most frequent expanded type for the POS of each paradigm generated for it. The ***GG*** baseline predicts for the unknown word the target type(s) used as features for this word in the classifier (e.g., for *Aufgabe*, these are the types from row (v) in Table 2). For the ***TnT*** baseline, we used the treebank available for the GG to train the TnT POS tagger (Brants, 2000) with the tagset consisting of all lexical types in the GG. The sentences extracted for each unknown word are then tagged in best-only mode. Similarly to the GG baseline, the unknown word is given the type(s) most fre-

quently assigned to it (80% threshold). The average type precision and type recall for the 400 test words are given in Table 5. Table 6 shows the F-measure per POS for the LA model.

Model	Prec(%)	Rec(%)	F-meas(%)
Naive	21.75	21.07	21.41
Naive POS	58.96	47.65	52.7
GG	55.03	57.12	56.06
TnT	61.21	49.17	54.53
LA with the GG	82.04	86.5	84.21

Table 5: Results on the 400 test words

POS	nouns	adj	verbs	adv
F-meas (%)	92.4	90.93	67.25	75.82

Table 6: F-measures per POS for the LA model

The LA model improves upon the baselines. The F-measure reaches 80% when 300 words are used for training and the learning curve flattens out at 1600 training words. The method performs very similar to the results reported for Dutch and Alpino, which demonstrates that it can be successfully applied outside the environment it was primarily developed for.

Predicting lexical entries for verbs is the hardest task for the LA model. The classifier has a strong bias towards assigning transitive and intransitive verb types. It either fails to predict infrequent frames or it wrongly predicts a transitive type for intransitive verbs and vice versa. Another difficulty for the model is the distinction which the GG makes between ergative and non-ergative verbs. The main issue with adverbs is that many of them can be used as adjectives as well. As a consequence, the classifier has a strong bias towards predicting an adverb type for words for which an adjective type has also been predicted. No pattern in the errors for nouns and adjectives can be identified.

6 Parsing Coverage and Accuracy

Having adapted the C&VN method to be used with the GG, we can now compare it with the supertagging methods by investigating how its application affects parsing coverage and accuracy on the FR corpus. We were not able to obtain the sentence sets used in Nicholson et al. (2008) and Cholakov et al. (2008), so we created a test set of 450 sentences randomly extracted from the FR corpus. All sentences contained at least one unknown word. The average sentence length is 17.16 tokens, with a ratio of 1.09 unknown words per sentence. For this experiment, we parse the 450 sentences with PET, under three conditions. In the first case, the standard configuration of the GG is used where the unknown word guesser assigns generic types to the unknown words. Dridan et al. (2008) report that passing a POS tagged input to the guesser enhances its performance and improves parsing coverage. That is why, in the second case, the 450 sentences were tagged with TnT outputting all POS tags with non-zero probabilities for each word. Then, the tagged sentences are parsed with PET. In the third case, we add to the GG lexicon lexical entries acquired offline by the adapted C&VN method.

The results are given in Table 7. The models employing POS tags and LA have practically the same coverage performance. The *GG+POS* one produced at least one parse for 113 sentences and the LA model produced analyses for those 113 sentences plus 4 more, thus giving a total of 117 parsed sentences. The standard GG model was able to cover 57 sentences. The parsed sentences for all models are manually examined and accuracy is again measured as the percentage of those parsed sentences which had a correct parse produced among the set of parses. There were 99 such sentences for the model which employs LA and 89 and 47 for the *GG+POS* and the *GG-standard* model, respectively. The drop in the accuracy for the *GG+POS* model compared to the standard GG one is consistent with the accuracy results reported in Dridan et al. (2008) for the ERG. We should note, though, that both the LA and *GG+POS* models also produced a correct parse for the 47 sentences for which the standard model delivered a correct analysis.

Model	Cov (%)	Acc (%)	LB Acc (%)
GG-standard	12.67	82.46	92.87
GG + POS	25.11	78.76	92.51
GG + LA	26	84.62	94.71

Table 7: Coverage and accuracy results for FR. *LB Acc* stands for labelled brackets accuracy.

The better accuracy result achieved by the LA model has to do mainly with the fact that the built-in guesser assigns noun types to the vast majority of the unknown words. The underspecified features in those entries create a lot of ambigu-

ity and make it harder for the parser disambiguation model to select the correct analysis. The LA method, on the other hand, supplies the parser with more detailed and linguistically accurate lexical entries which facilitates ambiguity resolution.

The results differ from the ones reported on the FR corpus for the suppertagging methods (Table 1) where the application of the tagger increased coverage but the price was lower accuracy. We attribute this to the bigger amount of features used in the classifier in the C&VN method which enabled the learning of more detailed lexical entries.

However, the estimation used up till now for accuracy is a rather crude one. Another way of looking at accuracy is to measure how close the top ranked parse for a given sentence is to the correct parse produced for this sentence. We selected the 47 sentences for which all three models have produced the correct parse and used them as our gold standard, to be able to report accuracy numbers, for the best parse.[6] Accuracy is measured in terms of labelled brackets. The results given in Table 7 show that not only the LA model produces a correct parse for more sentences but also the top ranked analysis is of better quality.

7 LA for Text Generation

As a further evaluation, we also investigate how the lexical entries acquired with LA affect sentence realisation. While in parsing the ambiguity in such less constrained lexical entries dissolves quickly in its context, there is a potential risk of overgeneration in the reverse process.

We selected 14 words assigned verb types by the tagger in the experiment with the FR corpus. Then, 64 sentences, each of which contains one of those 14 words, are extracted from corpora. We construct manually another sentence set where these words are replaced by verbs from the GG lexicon with a similar lexical type. This comparison set indicates what the performance of the GG would be with fully constrained, but otherwise similar lexical entries.

Realisation with the GG is performed within the LKB grammar engineering platform which provides a chart-based generator with various optimisations for packed parse forest (Carroll and Oepen, 2005). We parsed the two sentence sets and then used the best analysis to generate a realisation for each sentence. There were 3.28 realisations per sentence for the test set versus 3.16 for the comparison one. As for accuracy, a realisation is considered correct if it is an exact match of the original sentence (excluding punctuation). Despite the higher number for realisations per sentence for the test set, the quality of the realisations is the same for both sets– for 60 sentences a correct realisation is produced. Thus, the entries acquired with LA can be employed for both parsing and realisation.

We should note that realisation with the generic lexical entries acquired by the unknown word guesser of the parser is computationally not feasible because of the many underspecified features in those entries. Nicholson et al. (2008) and Cholakov et al. (2008) did not perform any experiments on realisation but we assume that the underspecification in the entries those methods produce would also make sentence realisation practically impossible.

8 Conclusion

Two types of methods for dealing with unknown words were compared. The application of methods, where a tagger was used to predict lexical descriptions for words unknown to the GG grammar of German, led to an increase in parsing coverage on a German newspaper corpus. However, accuracy was below the baseline, that is the accuracy of the standard GG setup. The other type of methods employ LA techniques to extend the lexicon of a grammar with new lexical entries for unknown words. We adapted one such method, namely the one of Cholakov and van Noord (2010), which has been primarily developed for the Dutch Alpino grammar, to be used with the GG. The application of the method led to an increase in both parsing coverage and accuracy on the same newspaper corpus. We attributed the better performance of the C&VN technique to the fact that it had access to more features during prediction, including such which came directly from the grammar, and it was able to assign more linguistically accurate entries to the unknown words. Last, in a smaller experiment, we showed that those entries can be successfully used for sentence realisation.

References

Tim Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of*

[6] That is why, the reported accuracy numbers are rather high.

the ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition, Ann Arbor, USA.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. In *Computational Linguistics*, volume 25(2), pages 237–65.

Petra Barg and Markus Walther. 1998. Processing unknown words in HPSG. In *Proceedings of the 36th Conference of the ACL*, Montreal, Quebec, Canada.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of EMNLP 2006*, Sidney, Australia.

Thorsten Brants. 2000. TnT– a statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing ANLP-2000*, Seattle, WA.

Ulrich Callmeier. 2000. PET– a platform for experimentation with efficient HPSG processing techniques. In *Journal of Natural Language Engineering*, volume 6, pages 99–107. Cambridge University Press.

John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP 2005)*, pages 165–176, Jeju Island, Korea.

Kostadin Cholakov and Gertjan van Noord. 2009. Combining finite state and corpus-based techniques for unknown word prediction. In *Proceedings of the 7th Recent Advances in Natural Language Processing (RANLP) conference*, Borovets, Bulgaria.

Kostadin Cholakov and Gertjan van Noord. 2010. Acquisition of unknown word paradigms for large-scale grammars. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-2010)*, Beijing, China.

Kostadin Cholakov, Valia Kordoni, and Yi Zhang. 2008. Towards domain-independent deep linguistic processing: Ensuring portability and re-usability of lexicalised grammars. In *Proceedings of COLING 2008 Workshop on Grammar Engineering Across Frameworks (GEAF08)*, Manchester, UK.

Stephen Clark and James Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'2004)*, Geneva, Switzerland.

Stephen Clark. 2002. Supertagging for combinatory categorical grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammar and Related Frameworks*, Venice, Italy.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC 2000*, Athens, Greece.

Berthold Crysmann. 2003. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP 2003*, Borovets, Bulgaria.

James Cussens and Stephen Pulman. 2000. Incorporating linguistic constraints into inductive logic programming. In *Proceedings of the Fourth Conference on Computational Natural Language Learning*.

Rebecca Dridan, Valia Kordoni, and Jeremy Nicholson. 2008. Enhancing perfomrmance of lexicalised grammars. In *Proceedings of ACL-08: HLT*, Columbus, Ohio.

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

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

Adam Kilgarriff and G Grefenstette. 2003. Introduction to the special issue on the web as a corpus. In *Computational Linguistics*, volume 29, pages 333–347.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th conference on Natural Language Learning (CoNLL-2002)*, pages 49–55, Taipei, Taiwan.

Jeremy Nicholson, Valia Kordoni, Yi Zhang, Timothy Baldwin, and Rebecca Dridan. 2008. Evaluating and extending the coverage of HPSG grammars: A case study for German. In *Proceedings of LREC-2008*, Marakesh, Morocco.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Melanie Siegel and Emily Bender. 2002. Efficient deep processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization*, Taipei, Taiwan.

Gertjan van Noord. 2006. At last parsing is now operational. In *Proceedings of TALN*, Leuven, Belgium.

Yi Zhang and Valia Kordoni. 2006. Automated deep lexical acquisition for robust open text processing. In *Proceedings of the Fifth International Conference on Language Resourses and Evaluation (LREC 2006)*, Genoa, Italy.

Yaozhong Zhang, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. A simple approach for HPSG supertagging using dependency information. In *Proceedings of 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'10)*, Los Angeles, CA.