Using Unknown Word Techniques To Learn Known Words

Kostadin Cholakov
University of Groningen
The Netherlands
k.cholakov@rug.nl

Gertjan van Noord
University of Groningen
The Netherlands
g.j.m.van.noord@rug.nl

Abstract

Unknown words are a hindrance to the performance of hand-crafted computational grammars of natural language. However, words with incomplete and incorrect lexical entries pose an even bigger problem because they can be the cause of a parsing failure despite being listed in the lexicon of the grammar. Such lexical entries are hard to detect and even harder to correct.

We employ an error miner to pinpoint words with problematic lexical entries. An automated lexical acquisition technique is then used to learn new entries for those words which allows the grammar to parse previously uncovered sentences successfully.

We test our method on a large-scale grammar of Dutch and a set of sentences for which this grammar fails to produce a parse. The application of the method enables the grammar to cover 83.76% of those sentences with an accuracy of 86.15%.

1 Introduction

In this paper, we present an automated two-phase method for treating *incomplete* or *incorrect* lexical entries in the lexicons of large-scale computational grammars. The performance of our approach is tested in a case study with the wide-coverage Alpino grammar (van Noord, 2006) of Dutch. When applied to real test sentences previously not covered by Alpino, the method causes a parsing coverage of 83.76% and the accuracy of the delivered analyses is 86.15%.

The main advantage of our approach is the successful combination of efficient *error mining* and *lexical acquisition* techniques. In the first phase, error mining pinpoints words which are listed in the lexicon of a given grammar but which nevertheless often lead to a parsing failure. This indicates that the current lexical entry for such a word is either wrong or incomplete and that one or more correct entries for this word are missing from the lexicon. Our idea is to treat the word as if it was unknown and, in the second phase, to employ lexical acquisition (LA) to learn the missing correct entries.

In the case study presented here, we employ the iterative error miner of de Kok et al. (2009). Since it has to be run on a large parsed corpus, we have parsed the Flemish Mediargus corpus ($\sim$1.5 billion words) with Alpino. The reason for this choice is the relatively large lexical difference between standard Dutch and Flemish. This increases the chance to encounter words which are used in Flemish in a way not handled by Alpino yet.

For example, the word *afwater* (to drain) is listed as a first person singular present verb in the Alpino lexicon. However, the error miner identifies this word as the reason for the parsing failure of 9 sentences. A manual examination reveals that the word is used as a neuter noun in these cases– *het afwater* (the drainage). Since there is no noun entry in the lexicon, Alpino was not able to produce full-span analyses.

After the error miner identifies *afwater* as a problematic word, we employ our machine learning based LA method presented in Cholakov and van Noord (2010) to learn new entries for this word. This method has already been successfully applied to the task of learning lexical entries for unknown words and, as the error miner, it can be used 'out of the box'. LA correctly predicts a neuter noun en-

Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 902–912,
MIT, Massachusetts, USA, 9-11 October 2010.

try for *afwater* and the addition of this entry to the lexicon enables Alpino to cover the 9 problematic sentences from the Mediargus corpus.

It should be noted that since our approach cannot differentiate between incomplete and incorrect entries, no entry in the lexicon is modified. We simply add the lexical entries which, according to the LA method, are most suitable for a given problematic word and assume that, if these entries are correct, the grammar should be able to cover previously unparsable sentences in which the word occurs.

The remainder of the paper is organised as follows. Section 2 describes the error miner. Section 3 presents the Alpino grammar and parser and the LA technique we employ. Section 4 describes an experiment where error mining is performed on the Mediargus corpus and then, LA is applied to learn new lexical entries for problematic words. Section 5 discusses the effect which the addition of the new entries to the lexicon has on the parsing coverage and accuracy. Section 6 provides a comparison between our approach and previous work similar in nature. This section also discusses the application of our method to other systems and languages as well as some ideas for future research.

2 Error Mining

The error miner of de Kok et al. (2009) combines the strengths of the error mining methods of van Noord (2004) and Sagot and de la Clergerie (2006). The idea behind these methods is that grammar errors lead to the parsing failure of some grammatical sentences. By running the grammar over a large corpus, the corpus can be split into two subsets– the set of sentences which received a full-span parse and the set of sentences failed to parse. Words or n-grams which occur in the latter set have a suspicion of being the cause of parsing failures.

van Noord (2004) defines the suspicion of a word sequence as:

$$(1) \qquad S(w_i...w_j) = \frac{C(w_i...w_j|error)}{C(w_i...w_j)}$$

where $C(w_i...w_j)$ is the number of sentences which the sequence $w_i...w_j$ occurs in and $C(w_i...w_j|error)$ is the number of occurrences of the sequence in unparsable sentences.

While this method performs well in identifying words and n-grams that are unambiguously suspicious, it also assigns incorrectly a high suspicion to forms which happen to occur often in unparsable sentences by 'bad luck'. The iterative error mining algorithm of Sagot and de la Clergerie (2006) tackles this problem by taking the following into account:

- If a form occurs within parsable sentences, it becomes less likely for it to be the cause of a parsing failure.
- The suspicion of a form depends on the suspicions of the other forms in the unparsable sentences it occurs in.
- A form observed in a shorter sentence is initially more suspicious than a form observed in a longer one.

However, because of data sparseness problems, this method is only able to handle unigrams and bigrams. Another potential problem is the absence of criteria to determine when to use unigrams and when bigrams to represent forms within a given sentence. Consider the trigram w_1, w_2, w_3 where w_2 is the cause of a parsing failure. In this case, the whole trigram as well as the bigrams w_1, w_2 and w_2, w_3 will become suspicious which would prevent the unigram w_2 from 'manifesting' itself.

To avoid this problem, de Kok et al. (2009) uses a preprocessor to the iterative miner of Sagot and de la Clergerie (2006) which iterates through a sentence of unigrams and expands unigrams to longer n-grams when there is evidence that this is useful. A unigram w_1 is expanded to a bigram w_1, w_2 if this bigram is more suspicious than both of its unigrams. The general algorithm is that the expansion to an n-gram $i...j$ is allowed when the following two conditions are fulfilled:

$$(2) \qquad \begin{aligned} S(i...j) &> S(i...j-1) \cdot expFactor \\ S(i...j) &> S(i+1...j) \cdot expFactor \end{aligned}$$

Within the preprocessor, suspicion is defined as shown in (1) and the *expFactor* is a parameter specially designed to deal with data sparseness.

As the error mining technique of de Kok et al. (2009) successfully overcomes the problems which

the other error mining methods we discussed encounter, we have chosen to employ this technique in our experiment.

3 Automated Lexical Acquisition

3.1 The Alpino Grammar and Parser

Since we employ Alpino for the purposes of our case study, it is convenient to explain the LA method we have chosen to use in the context of this system.

The Alpino wide-coverage parser is based on a large stochastic attribute value grammar. The grammar takes a 'constructional' approach, with rich lexical representations stored in the lexicon and a large number of detailed, construction specific rules (about 800).

Currently, the lexicon contains over 100K lexical entries and a list of about 200K named entities. Each word is assigned one or more lexical types. For example, the verb *amuseert* (to amuse) is assigned two lexical types– *verb(hebben,sg3,intransitive)* and *verb(hebben,sg3,transitive)*– because it can be used either transitively or intransitively. The other type features indicate that it is a present third person singular verb and it forms perfect tense with the auxiliary verb *hebben*.

3.2 Learning Algorithm

The goal of the LA method we describe Cholakov and van Noord (2010) is to assign correct lexical type(s) to a given unknown word.

It takes into account only open-class lexical types: nouns, adjectives and verbs. The types considered in the learning process are called *universal types*[1].

For a given word, a maximum entropy (ME) based classifier takes various morphological and syntactic features as input and outputs a ranked list of lexical types. The probability of a lexical type *t*, given an unknown word and its context *c* is:

$$(3) \qquad p(t|c) = \frac{exp(\sum_i \Theta_i f_i(t,c))}{\sum_{t' \in T} exp(\sum_i \Theta_i f_i(t',c))}$$

where $f_i(t,c)$ may encode arbitrary characteristics of the context and $< \Theta_1, \Theta_2, ... >$ is a weighting parameter which maximises the entropy and can be evaluated by maximising the pseudo-likelihood on a training corpus (Malouf, 2002).

[1]The adjectives can be used as adverbs in Dutch and thus, the latter are not considered to be an open class.

Features
i) a, af, afw, afwa
ii) r, er, ter, ater
iii) particle_yes #in this case *af*
iv) hyphen_no
v) noun⟨het,sg⟩, verb⟨sg1⟩
vi) noun(het,count,sg), noun(de,count,pl)
vii) noun(het), noun(count), noun(sg), noun(de) noun(pl)

Table 1: Features for *afwater*

Table 1 shows the features for *afwater*, the word we discussed in Section 1. Row **(i)** contains 4 separate features derived from the prefix of the word and 4 other suffix features are given in row **(ii)**. The two features in rows **(iii)** and **(iv)** indicate whether the word starts with a particle and if it contains a hyphen, respectively.

Further, the method we describe in Cholakov and van Noord (2009) is applied to generate the paradigm(s) of each word in question. This method uses a finite state morphology to generate possible paradigm(s) for a given word. The morphology does not have access to any additional 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 determine the correct paradigm(s). The web search heuristics are also able to determine the correct definite article (*de* or *het*) for words with noun paradigms.

One verb and one noun paradigm are generated for *afwater*. In these paradigms, *afwater* is listed as a first person singular present verb form and a singular *het* noun form, respectively. This information is explicitly used as features in the classifier which is shown in row **(v)** of Table 1.

Next, syntactic features for *afwater* are obtained by extracting a number of sentences which it occurs in from large corpora or Internet. These sentences are parsed with a different 'mode' of Alpino where this word is assigned all universal types, i.e. it is treated as being maximally ambiguous. For each sentence only the parse which is considered to be the best by the Alpino statistical disambiguation model

is preserved. Then, the lexical type that has been assigned to *afwater* in this parse is stored. During parsing, Alpino's POS tagger (Prins and van Noord, 2001) keeps filtering implausible type combinations. For example, if a determiner occurs before the unknown word, all verb types are typically not taken into consideration. This heavily reduces the computational overload and makes parsing with universal types computationally feasible.

When all sentences have been parsed, a list can be drawn up with the types that have been used and their frequency:

(4) noun(het,count,sg) 54
noun(de,count,pl) 7
tmp_noun(het,count,sg) 4
adjective(no_e(adv)) 4
proper_name(sg,'ORG') 1

The lexical types assigned to *afwater* in at least 80% of the parses are used as features in the classifier. These are the two features in row **(vi)** of Table 1. Further, as illustrated in row **(vii)**, each attribute of the considered types is also taken as a separate feature.

After the classifier predicts lexical types for each word, these predictions are subject to two additional steps of processing. In the first one, the generated word paradigms are explicitly used as a filtering mechanism. When a word is assigned a verb or an adjective type by the classifier but there is no verb or adjective paradigm generated for it, all verb or adjective predictions for this word are discarded.

The output of this 'filtering' is further processed in the second step which deals with the correct prediction of subcategorization frames for verbs. Following the observations made in Korhonen et al. (2000), Lapata (1999) and Messiant (2008), Cholakov and van Noord (2010) employ a maximum likelihood estimate (MLE) from observed relative frequencies with an empirical threshold to filter out low probability frames.

Since some frames could be very infrequent and the MLE method may not capture them, the generated word paradigms are used to increase the number of contexts observed for a given verb. Additional sentences are extracted for each form in the paradigm of a given word predicted to be a verb. These sentences are again parsed with the universal types. Then we look up the assigned universal verb types, calculate the MLE for each subcategorization frame and filter out frames with MLE below some empirical threshold.

4 Learning New Lexical Entries

Before we start with the description of the experiment, it is important to note that Alpino is very robust– essentially, it always produces a parse. If there is no analysis spanning the whole sentence, the parser finds all parses for each substring and returns what it considers to be the best sequence of non-overlapping parses. However, in the context of this experiment, a sentence will be considered successfully parsed only if it receives a full-span analysis. For the sake of clarity, from now on we shall use the terms *coverage* and *cover* only with regard to such sentences. The term *parsing failure* shall refer to a sentence for which Alpino fails to produce a full-span analysis.

4.1 Error Mining on Mediargus

The first step in our experiment is to perform error mining on the Mediargus corpus. The corpus consists of texts from Flemish newspapers from the period between 1998 and 2007. It contains about 1.5 billion words (∼78M sentences). The corpus has been parsed with Alpino and the parsing results are fed into the error miner of de Kok et al. (2009). The parser has not produced a full-span analysis for 7.28% of the sentences (∼5.7M sentences).

When finished, the error miner stores the results in a data base containing potentially problematic n-grams. Each n-gram is linked to its suspicion score and the sentences which it occurs in and which were not covered by Alpino.

Before proceeding with LA, however, we should identify the n-grams which are indicative for a problem in the lexicon. The first step in this direction is to extract all *unigrams* from the data base which have a suspicion equal to or greater than 0.7 together with the uncovered sentences they occur in. This resulted in a list containing 4179 unique unigrams. Further, we select from this list only those unigrams which have lexical entries in the Alpino lexicon and occur in more than 5 sentences with no full-span

parse. Sometimes, the error miner might be wrong about the exact word which causes the parsing failure for a given sentence. The 5 sentences empirical threshold is meant to guarantee that the selected words are systematically causing problems for the parser.

The result of this selection is 36 unigrams (words) which occur in a total of 388 uncovered sentences– an average of 10.78 sentences per word. The small number of selected words is due to the fact that most of the problematic 4179 unigrams represent tokenization errors (two or more words written as one) and spelling mistakes which, naturally, are not listed in the Alpino lexicon. Very few of the 4179 unigrams are actual unknown words. Table 2 shows some of the problematic unigrams and their suspicions.

opVorig	0.898989
GentHoewel	0.89759
Nieuwpoortl	0.897414
SportTijdens	0.897016
DirvenDe	0.896428
mistrap	0.896038
Dwoeurp	0.896013
passerde	0.89568
doorHugo	0.893901
goedkmaken	0.892407
ManneN	0.891539
toegnag	0.891523

Table 2: Problematic unigrams and their suspicions

It can be seen immediately that most of the unigrams presented in the table are tokenization errors. There are also some typos. The unigram *passerde* should be written as *passe**e**rde*, the past singular verb form of the verb 'to pass' and *toegnag* is the misspelled noun *toegang* (access). The only problematic unigram with a lexical entry in the Alpino lexicon is *mistrap* (misstep, to misstep).

Although the experiment setup yields a small test set, we employ it because the words in this set represent 'clear-cut' cases. This allows us to demonstrate better the effect of our technique.

4.2 Applying Lexical Acquisition

Our assumption is that incomplete or incorrect lexical entries prevented the production of full-span parses for the 388 sentences in which the 36 problematic words pinpointed by the error miner occur. That is why, in the second step of the experiment, these words are temporarily removed from the Alpino lexicon, i.e. they are treated as unknown words, and we employ the LA method presented in the previous section to learn offline new lexical entries for them.

The setup for the learning process is exactly the same as in Cholakov and van Noord (2010). The set of universal types consists of 611 types and the ME-based classifier has been trained on the same set of 2000 words as in Cholakov and van Noord (2010). Those types predicted by the classifier which account together for less than 5% of probability mass are discarded.

In order to increase the number of observed contexts for a given word when parsing with the universal types, up to 100 additional sentences in which the word occurs are extracted from Internet. However, when predicting new lexical entries for this word, we want to take into account only sentences where it causes a parsing failure. It is in such sentences where a new lexical entry can be learnt through LA. For example, the LA method would be able to predict a noun entry for *afwater* if it focuses only on contexts where it has a noun reading, i.e. on sentences not covered by Alpino.

That is why, the sentences we extracted from Internet are first parsed with the standard Alpino configuration. When averaging over the 36 sentence sets, it turns out that Alpino has been able to cover 10.05% of the sentences. Although we cannot be sure that the 36 words are the cause of a parsing failure in each of the uncovered sentences, this low coverage indicates once more that Alpino has systematic problems with sentences containing these words.

Then, the uncovered sentences from Internet together with the 388 problematic sentences from the Mediargus corpus are parsed with Alpino and the universal types. For example, the list of universal types assigned to *afwater* in (4) contains mostly noun types, i.e. the kind of types which are currently not in the lexicon for this word and which we want to learn.

The result of the LA process is the prediction of a total of 102 lexical types, or 2.83 types per word. This high number is due to the fact that 25 words receive verb predictions. Since a verb can have vari-

ous subcategorization frames, there is one type assigned for each frame. For example, *inscheppen* (to spoon in(to)) receives 3 types which differ only in the subcategorization frame– *verb(hebben,inf,tr.)*, *verb(hebben,inf,intr.)* and *verb(hebben,inf,np_np)*. However, the infinitive in Dutch is also the form for plural present and *inscheppen* correctly receives 3 more predictions– *verb(hebben,pl,tr.)*, *verb(hebben,pl,intr.)* and *verb(hebben,pl,np_np)*.

Let us examine the most frequent types of lexicon errors for the 36 problematic words by looking at the current Alpino lexical entries for some of these words and the predictions they receive from the LA method. The original Alpino entries for 19 of the 25 words predicted to be verbs are a product of a specific lexical rule in the grammar. Consider the following sentences:

(5) a. Ik schep de soep in de kom.
I spoon the soup in the bowl
'I spoon the soup into the bowl.'

b. dat ik de soep de kom in schep
that I the soup the bowl in spoon
'that I spoon the soup into the bowl'

c. dat ik de soep de kom *in*schep
that I the soup the bowl in spoon
'that I spoon the soup into the bowl'

We see in (5-b) that the preposition *in* is used as a postposition in the relative clause. However, in such cases, there is linguistic evidence that *in* behaves as a separate verb particle. That is why, as shown in (5-c), people sometimes write *in* and the verb together when they occur next to each other in the sentence. To account for this, Alpino employs a special lexical rule. This rule assigns a certain type of subcategorization frame to verbs like *inscheppen* where a postposition can be interpreted as a separable particle. That subcategorization frame requires a noun phrase ('the soup' in (5-c)) and a locative NP ('the bowl' in (5-c)).

However, in some cases, the entries generated by this lexical rule cannot account for other possible usages of the verbs in question. For example,

(6) U moet deze zelf inscheppen.
you must this yourself spoon in.INF
'You have to spoon this in yourself.'

Alpino fails to parse this sentence because *inscheppen* is used without a locative NP. Now, when the LA method has predicted a transitive verb type for *inscheppen*, the parser should be able to cover the sentence. Other such examples from our data include ***weg****wist* (to erase.3PER.SG), ***onder****ligt* (to lie under.3PER.SG), etc.

Further, there are 10 words, including *afwater*, which represent cases of nominalisation currently not accounted for in the Alpino lexicon. The LA process correctly predicts noun types for these words. This should enable the parser to cover sentences like:

(7) Die moet een deel van het afwater vervoeren.
this must a part from the drainage transport/move
'This has to move a part of the drainage.'

where *afwater* is used as a noun.

There are also 3 words which correctly receive adjective predictions. Currently, their lexical entries are incomplete because they are assigned only past participle types in the lexicon. However, past participles in Dutch can also act as adjectives. For historical reasons, this systematic ambiguity is not treated as such in Alpino. Each participle should also have a separate adjective lexical entry but, as we see, this is not always the case.

5 Results

After LA is finished, we restore the original lexical entries for the 36 words but, additionally, each word is also assigned the types which have been predicted for it by the LA method. The 388 problematic sentences from the Mediargus corpus are then re-parsed with Alpino. We are interested in observing:

1. how many sentences receive a full-span analysis

2. how the parsing accuracy of Alpino changes

Table 3 shows that when the Alpino lexicon is extended with the lexical entries we learnt through LA, the parser is able to cover nearly 84% of the sentences, including the ones given in (6) and (7). Since there is no suitable baseline which this result can be compared to, we developed an additional model which indicates what is likely to be the maximum coverage that Alpino can achieve for those sentences by adding new lexical entries only.

In this second model, for each of the 36 words, we add to the lexicon all types which were successfully used for the respective word during the parsing with universal types. In this way, Alpino is free to choose from all types it has considered suitable for a given word, i.e. the parser is not limited by the outcome of the LA process but rather by the overall quality of the grammar.

The 'universal types' model performs better than ours– it achieves 87.9% coverage. Still, the performance of our model is close to this result, i.e. close to what we consider to be the maximal possible coverage of Alpino for these 388 sentences when only LA is used.

Model	Coverage (%)
Our model (Alpino + LA)	83.76
Universal types	87.89

Table 3: Coverage results for the re-parsed 388 problematic sentences

Some of the sentences which cannot be covered by both models are actually not proper sentences but fragments which were wrongly identified as sentences during tokenization. Many other cases include sentences like:

(8) Een gele frommel papier, Arabische lettertekens.
a yellow crease paper Arabic characters
'A yellow paper crease, Arabic characters.'

which is probably the caption of a photo or an illustration. However, because of the absence of a verb, Alpino splits the analysis into two parts– the part before the comma and the part after the comma.

Here is a more interesting case:

(9) Als we ons naar de buffettafel begeven, mistrap ik
when we us to the buffet proceed misstep I
me.
myself
'When we proceed to the buffet I misstep.'

The LA method does not predict a reflexive verb type for *mistrap* which prevents the production of a full-span analysis because Alpino cannot connect the reflexive pronoun *me* to *mistrap*. In this case, however, the universal type model outperforms ours. A reflexive verb type is among the universal types and thus, Alpino is able to use that type to deliver a full-span parse. We should note though, that LA correctly predicts a noun type for *mistrap* which enables Alpino to parse successfully the other 14 sentences which this word occurs in.

Let us now look at the correctness of the delivered parses. To estimate the accuracy of the parser, we have randomly selected 100 sentences out of the 388 sentences in the test set and we have manually annotated them in order to create a gold standard for evaluation.

Accuracy in Alpino is measured in terms of dependency relations. The accuracy for sentences which are not assigned a full-span analysis but a sequence of non-overlapping parses can still be larger than zero because, within these parses, some correct dependency relations could have been produced. That is why, though the coverage of Alpino for the selected 100 sentences is zero, we can still obtain a number for accuracy and use it as a baseline for comparison. Clearly, this baseline is expected to perform worse than both our model and the universal types one since those are able to cover most of the sentences and thus, they are likely to produce more correct dependency relations. However, it gives us an idea how much extra quality is gained when coverage improves.

The accuracy results for the 100 annotated sentences are given in Table 4. The average sentence length is 18.9 tokens.

Model	Accuracy (%)	msec/sentence
Alpino	63.35	803
Our model	86.15	718
Universal types	85.12	721

Table 4: Accuracy results for the 100 annotated sentences

Our model achieves the highest accuracy without increasing the parse times. Further, the baseline has a much lower result which shows that coverage is not gained on the expense of accuracy.

Our model and the universal types one achieve the same accuracy for most of the sentences. However, the universal types model has an important disadvantage which, in some cases, leads to the production of wrong dependency relations. The model predicts a large number of lexical types which, in turn, leads to large *lexical ambiguity*. This lexical ambiguity increases the number of possible analyses Alpino chooses from, thus making it harder for the

parser to produce the correct analysis. Let us consider the following example where a sentence is covered by both models but the universal types model has lower accuracy:

(10) Dat wij het rechttrokken, pleit voor onze
that we it straighten.PAST.PL. plead for our
huidige conditie.
current condition
'It pleads for our condition that we straightened it.'

Here, *het* is the object of the verb *rechttrokken*. However, although there are transitive verb types among the universal types assigned to *rechttrokken*, Alpino chooses to use a verb type which subcategorizes for a measure NP. This causes for *het* to be analysed not as an object but as a measure complement, i.e. the produced dependency relation is incorrect.

The LA method, on the other hand, is much more restrictive but its predictions are also much more accurate. Since it considers sentences containing other forms of the paradigm of *rechttrokken* when predicting subcategorization frames, the LA method correctly assigns only one transitive and one intransitive verb type to this word. This allows Alpino to recognize *het* as the object of the verb and to produce the correct dependency relation.

The few cases where the universal types model outperforms ours include sentences like the one given in (9) where the application of our model could not enable Alpino to assign a full-span analysis. Sometimes, the LA method is too restrictive and does not output some of the correct types. These types, on the other hand, could be provided by the universal types model and could enable Alpino to cover a given sentence and thus, to produce more correct dependency relations. Allowing for the LA method to predict more types, however, has proven to be a bad solution because, due to the increased lexical ambiguity, this leads to lower parsing accuracy.

6 Discussion

6.1 Comparison to Previous Work

The performance of the technique we presented in this paper can be compared to the performance of a number of other approaches applied to similar tasks.

Zhang et al. (2006) and Villavicencio et al. (2007) use error mining to semi-automatically detect English multiword expressions (MWEs). Then, they employ LA to learn proper lexical entries for these MWEs and add them to the lexicon of a large-scale HPSG grammar of English (ERG; (Copestake and Flickinger, 2000)). This increases parsing coverage by 15% to 22.7% for a test set of 674 sentences containing MWEs and parsed with the PET parser (Callmeier, 2000). In both studies, however, the combination of error mining and LA is applied to a very specific task whereas our method is a general one.

Nicolas et al. (2008) employ a semi-automatic method to improve a large-scale morphosyntactic lexicon of French (Le*fff*; (Sagot et al., 2006)). The lexicon is used in two grammars– the FRMG (Thomasset and de la Clergerie, 2005), a hybrid Tree Adjoining/Tree Insertion Grammar, and the SxLFG-FR LFG grammar (Boullier and Sagot, 2006). The first step in this approach is also the application of an error miner (Sagot and de la Clergerie, 2006) which uses a parsed newspaper corpus (about 4.3M words) to pinpoint problematic unigrams.

The crucial difference with our method is in the second step. Nicolas et al. (2008) assign *underspecified* lexical entries to a given problematic unigram to allow the grammar to parse the uncovered sentences associated with this unigram. Then, these entries are ranked based on the number of successful parses they have been used in.

The use of underspecification, however, causes large ambiguity and severe parse overgeneration (observed also in Fouvry (2003)). As a consequence of that, the ranked list of lexical entries for each unigram is manually validated to filter out the wrong entries. The employment of LA in our approach, on the other hand, makes it fully automatic. The ranking of the predictions is done by the classifier and the predicted entries are good enough to improve the parsing coverage and accuracy without any manual work involved. Generally, recent studies (Baldwin, 2005; Zhang and Kordoni, 2006; Cholakov et al., 2008; Cholakov and van Noord, 2010) have clearly shown that when it comes to learning new lexical entries, elaborate LA techniques perform better and are more suitable for large-scale grammars than un-

derspecification[2].

Further, the naive ranking system used in Nicolas et al. (2008) puts a correctly generated entry for an infrequent usage of a given word (e.g., a verb with a rare subcat frame) in the bottom of the ranked list because of the low number of sentences in which this entry is used. The LA method we employ is more sensitive to rare usages of words because it considers occurrences of the word in question outside the parsed corpus (very important if the corpus is domain-specific) and it also takes into account all forms in the paradigm(s) of the word. This increases the chances of a rare usage of this word to 'manifest' itself.

Nicolas et al. (2008) uses the lexical entries which remain after the manual validation to re-parse the newspaper corpus. 254 words (mostly verbs) are corrected and the parse coverage increases by 3.4% and 1.7% for the FRMG and the SxLFG, respectively. However, the authors do not mention how many of the original uncovered sentences they are able to cover and therefore, we cannot compare our coverage result. Nothing is said about the parsing accuracy. Even with manually validated lexical entries, it is still possible for the grammar to produce full-span but wrong analyses.

6.2 Application to Other Systems and Languages

It is important to note that this paper should be viewed as a case study where we illustrate the results of the application of what we believe to be a good algorithm for dealing with incomplete or incorrect lexical entries– namely, the combination of error mining and LA. However, our method is general enough to be applied to other large-scale grammars and languages.

The error mining is directly usable as soon as there is a large parsed corpus available. The LA technique we employed is also quite general provided that certain requirements are fulfilled. First, words have to be mapped onto some finite set of labels of which a subset of open-class (universal) labels has to be selected. This subset represents the labels which can be predicted for unknown words. Second, we need a parser to analyse sentences in which a given unknown word occurs. Finally, the ME-based classifier allows for arbitrary combinations of features and therefore, any (language-specific) features considered useful can be included. As for the paradigm generation method, the idea of combining a finite state morphology and web heuristics is general enough to be implemented for different languages.

We have already started investigating the applicability of our method to the FRMG and a large-scale grammar of German and the initial experiment and results we have obtained are promising.

6.3 Future Research

Currently, our algorithm handles only unigrams (words). However, it would be useful to extend it, so it can work with longer n-grams. For example, a given word could have some reading which is not yet handled in the lexicon only within a particular bi- or trigram.

Consider the bigram '*schampte af*' which has been identified as problematic by the error miner. It represents the particle verb '*afschampte*' (to glance.PAST.SG). Although the lexicon contains a verb entry for '*schampte*', there is no entry handling the case when this verb combines with the particle '*af*'. Another example is the bigram '*de slachtoffer*' (the victim). In standard Dutch, the noun '*slachtoffer*' goes with the '*het*' definite article which is marked in its lexical entry. However, in Flemish it is used with the '*de*' article.

Our method is currently not able to capture these two cases since they can be identified as problematic on bigram level and not when only unigrams are considered.

Further, the definition of what the error miner considers to be a successful parse is a rather crude one. As we saw, even if the grammar is able to produce a full-span analysis for a given sentence, this analysis could still not be the correct one. Therefore, it is possible that a word could have a problematic lexical entry even if it only occurs in sentences which are assigned a full-span parse. Currently, such a word will not be identified as problematic by the error miner. That is why, some (statistical) model which is capable of judging the plausibility of a parse should be developed and incorpo-

[2]In Nicolas et al. (2008) the authors also admit that an elaborate LA technique will produce better results.

rated in the calculation of the suspicions during error mining.

References

Tim Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, Ann Arbor, USA.

Pierre Boullier and Benoît Sagot. 2006. Efficient parsing of large corpora with a deep LFG parser. In *Proceedings of LREC'06*, Genoa, Italy.

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.

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.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resource and Evaluation (LREC 2000)*, Athens, Greece.

Daniël de Kok, Jianqiang Ma, and Gertjan van Noord. 2009. A generalized method for iterative error mining in parsing results. In *Proceedigns of the 2009 Workshop on Grammar Engineering Across Frameworks, ACL-IJCNLP 2009*, pages 71–79, Singapore.

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.

Anna Korhonen, Genevieve Gorell, and Diana McCarthy. 2000. Statistical filtering and subcategorization frame acquisition. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong, China.

Mirella Lapata. 1999. Acquiring lexical generalizations from corpora. A case study for diathesis alternations. In *Proceedings of the 37th Annual Meeting of ACL*, Maryland, USA.

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.

Cedric Messiant. 2008. A subcategorization acquisition system for French verbs. In *Proceedings of the ACL 2008 Student Research Workshop*, Columbus, OH.

Lionel Nicolas, Benoît Sagot, Miguel Molinero, Jacques Farré, and Eric de la Clergerie. 2008. Computer aided correction and extension of a syntactic wide-coverage lexicon. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, pages 633–640, Manchester, UK.

Robbert Prins and Gertjan van Noord. 2001. Unsupervised POS-tagging improves parcing accuracy and parsing efficiency. In *Proceedings of IWPT*, Beijing, China.

Benoît Sagot and Eric de la Clergerie. 2006. Error mining in parsing results. In *Proceedings of the 44th Meeting of the Association for Computational Linguistics (ACL'06)*, pages 329–336, Morristown, NJ, USA.

Benoît Sagot, Lionel Clément, Eric de la Clergerie, and Pierre Boullier. 2006. The Lefff 2 syntactic lexicon for French. In *Proceedings of LREC'06*, Genoa, Italy.

François Thomasset and Eric de la Clergerie. 2005. Comment obtenir plus des méetagrammaires. In *Proceedings of TALN'05*, Dourdan, France.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 446–453, Barcelona, Spain.

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

Aline Villavicencio, Valia Kordoni, Yi Zhang, Marco Idiart, and Carlos Ramisch. 2007. Validation and evaluation of automatically acquired multiword expressions for grammar engineering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1034–1043, Prague, Czech Republic.

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.

Yi Zhang, Valia Kordoni, Aline Villavicencio, and Marco Idiart. 2006. Automated multiword expression prediction for grammar engineering. In *Proceedings of*

the ACL Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties, pages 36–44, Sydney, Australia.