Automatically Extending the Lexicon for Parsing

Tim van de Cruys
Alfa-informatica
University of Groningen
t.van.de.cruys@rug.nl

Abstract. This paper describes a method for automatically extending the lexicon of wide-coverage parsers. The method is an extension to the automatic detection of coverage problems of natural language parsers, based on large amounts of raw text (van Noord 2004). The goal is to extend grammar coverage, focusing in particular on the acquisition of lexical information for missing and incomplete lexicon entries (including subcategorization frames). In order to assign lexical entries for unknown words, or for words for which the lexicon only contains a subset of its possible lexical categories, we propose to apply a parser to a set of unannotated sentences containing the unknown word, or to a set of unannotated sentences (found by error mining) in which the word apparently was used with a missing lexical category. The parser will assign all universal lexical categories to the problematic word. Once the parser has found a result for the sentence, it can output the lexical category that was eventually used in its best parse. If this process is repeated for a large enough sample of sentences, it is expected that either a single or a small number of lexical categories can then be identified which are to be taken as the correct lexical categories of this word. A maximum entropy classifier is trained to select the correct lexical categories.

1 Introduction

Deep grammar parsing techniques have improved tremendously in the last few years. The emergence of adequate grammar descriptions and efficient parsing techniques have made it theoretically feasible to parse instances of raw text. The consequences of Moore's law (computational power doubles every 24 months) ensure that this is also practically feasible.

The main factor that needs to be improved is the *coverage*, also called the *robustness* of the parser. Hand-crafted linguistic descriptions such as wide-coverage grammars remain – despite the tremendous improvements – still quite incomplete. The hand-crafted lexicon is often the problem child. As previous work has noted (Baldwin, Bender, Flickinger, et al. 2004), most coverage problems are due to missing or incomplete lexicon entries. In this paper, a method is described that is able to automatically detect and correct those missing or incomplete entries. The method builds on the *error mining* technique described in van Noord (2004), that is able to automatically discover systematic mistakes in a parser by using very large but unannotated corpora. The technique is summarized in section 3.1.

In the experiments described here, the Alpino wide-coverage parser for Dutch is used (Bouma, van Noord, and Malouf 2001; van der Beek, Bouma, and van Noord 2002). This

Proceedings of the Eleventh ESSLLI Student Session
Janneke Huitink & Sophia Katrenko (editors)

parser is based on a large (±600 grammar rules) constructionalist Head-driven Phrase Structure Grammar (HPSG) for Dutch, as well as a very large (>100K words) lexicon. The parser is robust in the sense that it essentially always produces a parse. If a full parse is not possible for a given sentence, then the parser returns a (minimal) number of parsed nonoverlapping sentence parts.

2 Previous Work

2.1 Unification-based Grammar Induction

In unification-based frameworks such as HPSG, information in the lexicon may be underspecified, to become more specific only when it is actually used in a parse tree. An example is an English verb form like *drink*, which is only specified as 'present tense' (except for the third person singular). It is only when the form combines with a subject (such as the pronoun *I*) that it can be fully specified ('present tense first person singular'). This property of unification-based frameworks can be used to induce the grammatical features of unknown words. The feature structure of an unknown word is incrementally updated, as more and more sentences with different occurrences of the unknown word form are read by the unification algorithm. This kind of technique is elaborated in Erbach (1990) and Barg and Walther (1998). Fouvry (2003) applies the technique to a large-coverage grammar for German.

A problem with this approach is that the feature structure will be partly too general and partly too specific. Barg and Walther (1998) discuss in this view the concepts of generalisable and revisable information. The former are values that are too specific, while the latter are values that may be overwritten. The algorithm as such also is not able to cope with errors in text: it doesn't take any statistics into account.

2.2 Statistical Grammar Induction

Next to the more rigid framework of inducing feature structures, quite some authors have taken a statistical approach in lexicon induction. One of the first efforts to incorporate statistics in order to induce lexicon entries (and more specifically subcategorization frames) has been made by Brent (1993). His work makes use of statistics (more specifically hypothesis testing on binomial frequency data) to prevent noise from blurring the results. However, his work does not rely on full parsing of sentences, but on certain 'morpho-syntactic cues', to discover verbs and their arguments.

Schulte im Walde (2002) also pursued a statistical approach in inducing subcategorization frames, by making use of a probabilistic context free grammar (PCFG). The context free grammar yields for each verb a frequency distribution of subcategorization frames; applying a cut-off yields the subcategorization frames found by the algorithm.

The approach described in this paper differs from the previous approaches in two ways:

- the use of a broad-coverage deep grammar parser in order to find the possible lexical types of an unknown word;
- the use of a maximum entropy classifier to classify the output of the parsing method, in order to find the actual lexical types.

3 Methodology

3.1 Error Mining

The *error mining* technique assumes that a large corpus of sentences is available. Each sentence is a sequence of tokens (including words as well as punctuation marks, etc.). The parser is run on all sentences, and we note for which sentences the parser is successful[1]. The parsability of a word is defined as the ratio of the number of times the word occurs in a sentence with a successful parse and the total number of sentences that this word occurs in. Thus, if a word only occurs in sentences that cannot be parsed successfully, the parsability of that word is 0. On the other hand, if a word only occurs in sentences with a successful parse, its parsability is 1. If there is no reason to believe that a word is particularly easy or difficult, then we expect its parsability to be equal to the coverage of the parser (the proportion of sentences with a successful parse). If its parsability is (much) lower, then this indicates that something is wrong. Normally, the coverage of the parser lies between 91% and 95%. Yet, for *many* words, parsability values were found much lower than that, including quite a number of words with parsability 0.

Words or word sequences that are found by this technique are considered problematic. This problem might be due to missing grammatical constructions, or due to missing or incomplete lexical entries. This paper focuses on the latter.

3.2 Automatic Lexical Acquisition Algorithm

Parsing with Universal Tagset

Words that have been found problematic by the error mining technique may then be fed to the lexical acquisition algorithm. For each problematic word, a large number of sentences (in our experiments, we used 100) containing the word is extracted from large corpora, or taken from the Internet. Those sentences are parsed with a different version of Alpino: a parsing method has been used in which all possible 'universal tags' are assigned to the unknown or problematic word. By universal tags, we mean all tags that belong to an open part of speech class.[2] Infrequent tags and function word tags are not taken into consideration. This boils down to a universal tagset of 340 tags. Note that 'tag' in this sense does not indicate the part of speech tag, but the atomic lexical type used by

[1]In the scope of this paper, a successful parse means a parse that spans the whole sentence.

[2]If a certain tag appears with more than 15 different words in a large corpus, it is considered universal. The outcome has been slightly manually adapted to get a systematic tagset.

a parser. These lexical types include grammatical information such as subcategorization frames.[3] The parser assigns all universal tags to the unknown word, and each parse that is formed this way receives a probability score by the parser's disambiguation component.[4]

During the parsing process, Alpino's POS-tagger (Prins and van Noord 2001) keeps filtering implausible tag combinations. For example, if a determiner appears in front of the unknown word, the extensive range of verb tags should not be taken into consideration. The POS-tagger makes sure these tags will be filtered out. This process heavily reduces the computational overload, and makes the parsing method computationally feasible.

The parse that is considered the best parse by the disambiguation model is preserved. The tag that has been assigned in the best parse is our best guess given the actual sentence. When all sentences have been parsed, a list can be drawn up with the tags that have been used successfully, and their frequency. The correct tag(s) can then be determined statistically.

Take the following example sentences, in which the unknown word *sneup* appears (tags have been somewhat simplified to improve readability):

(1) Ik heb zin in sneup.
I feel like sneup
I would like some sneup.

(2) De sneup ligt in de kast.
The sneup lies in the cupboard
The sneup is in the cupboard.

(3) Ik wil sneup!
I want sneup
I want sneup!

- **ik**/pronoun(1st,sg,nom) **heb**/verb(hebben,sg1,transitive) **zin**/noun(de,sg) **in**/preposition(in) **sneup**/***noun(de,sg)***
- **de**/determiner(de) **sneup**/***noun(de,sg)*** **ligt**/verb('hebben/zijn',sg3,ld_pp) **in**/preposition(in) **de**/determiner(de) **kast**/noun(de,sg)
- **ik**/pronoun(1st,sg,nom) **wil**/verb(hebben,modal,intransitive) **sneup**/***adverb***

In these examples, the parser assigns twice the tag 'noun(de,sg)' to the word *sneup* and once the tag 'adverb'. This makes the first tag the most probable.

[3] An example of such a lexical type tag is **verb(hebben,inf,transitive)**. This tag indicates an infinitival form of a verb combining with the auxiliary *hebben*, used transitively.

[4] Alpino's disambiguation component uses a maximum entropy model. This component is explained in detail in van Noord (2006).

Adapting the Disambiguation Component

For the parsing method to work properly, the statistical disambiguation model of the parser had to be adapted. The parser's disambiguation model heavily relies on the lexicon. Based on training data, the disambiguation model has, for example, a preference to parse prepositional phrases as a prepositional complement to the verb, if such subcategorization frame exists. But this doesn't make any sense when parsing with a universal tagset, as every prepositional phrase would get classified as a complement to the verb. This problem is overcome by weighting each universal tag by its a priori probability (the actual frequency with which the tag appears in the training data). This way, the probability that a subcategorization frame with a prepositional complement is selected for an unknown verb, is scaled to the overall probability that such a subcategorization frame actually appears. In general, when the tag of a certain word in the sentence is unknown, we want the disambiguation component to take into account the a priori probability of all tags in the tagset when assigning a certain tag to the unknown word.

Word Classification

A maximum entropy (maxent) classifier (Le 2004) has been trained to classify the unknown words, taking as features the outcomes of the procedure described (i.e. the tags that were successful). To enable the maxent classifier to make broader generalizations, each subattribute of the tags (e.g. singular/plural difference with nouns, adjective attributes, subcategorization frame of verbs, ...) has also been handed to the classifier separately.

Next to the information yielded by our parsing method, the morphology of the unknown word is taken into account. The morphology features that are used are:

- the word ending (three last letters, two last letters, last letter)
- +/- past participle, e.g. ***ge****fiets****t***
- +/- word starts with particle, e.g. ***rond****fietsen*

A finite state automaton has been designed to determine whether a word has the characteristics of a past participle.

Training the Classifier

To be able to work with decent training data, Alpino has been 'untrained' for a small part of the lexicon (about 1500 words, of which 1000 words have been used for training). Those words were considered unknown by Alpino, yielding a list of possible tags assigned by the procedure described above. The correct tags, to be used for training, could be extracted from the original lexicon.

As in many natural language processing applications, ambiguity is an important problem. This is no different when trying to deduce a word's possible tags. Part of this ambiguity can be handled by the algorithm, as some words are consistently ambiguous (i.e. the kind of ambiguity that would be described by lexical rules). An example of such ambiguity is to be found in the Dutch verb system: the infinitive verb form in Dutch will also always be the plural form of the verb. This kind of systematic ambiguity is handled by the algorithm: if a structurally ambiguous tag has been found by the algorithm, the other one is automatically added by a number of manually constructed rules.

But the majority of ambiguity is not that straightforward. Such ambiguity is a potential problem for the algorithm. Consider the following sentences:

(4) 's Zomers fiets ik graag.
in summer bike.V I like
In summer, I like to bike.

(5) Ik rijd graag rond op mijn fiets.
I ride like around on my bike.N
I like to ride around on my bike.

Sentences 4 and 5 are an example of the latter kind of ambiguity: in 4, *fiets* is used as the first person singular of the verb *fietsen.* In 5, *fiets* is a noun. Note that there is quite some ambiguity of this kind, as it does not only depend on homonymous words, but also on internal ambiguity such as different subcategorization frames. In training, this kind of ambiguity is tackled in the following way: if a certain word had more than one lexical type in the lexicon, the classifier has been trained with all the lexical types available for the same attributes. In other words, if a word is ambiguous (as stipulated by the lexicon), both tags are taken into account for each context.

4 Results & Evaluation

4.1 Results

The classifier yields a probability score for each tag. Only the best tags are kept, which are the tags with a probability higher than or equal to 6.5%. The best tags are also determined relatively. If the probability score of a certain tag divided by the score of the next best tag on the list is higher than 8, the next tags are not taken into consideration. These values are not chosen randomly: they yield the highest f-score for the development data.

To evaluate the classifier, the same procedure has been used as for training: words have been parsed with a version of Alpino that has been untrained for these words, and the results have been compared with the tags that are available in the original lexicon. As one word might have several lexical types, we have evaluated the results in terms of precision and recall. Precision indicates how many of the lexical types that have been found by our algorithm are correct. Recall indicates how many of the lexical types of a certain word are actually found. The results given are the average precision and recall for the ± 500 test words.[5]

[5]Note that this evaluation framework evaluates precision and recall quite strictly: if one of the subat-

Two kinds of baseline have been used:

- a naive baseline in which every unknown word is assigned the overall most frequent tag, namely [noun(de,sg)];
- a more elaborate baseline, in which the most frequent tag for the part of speech of the unknown word is assigned.[6] The most frequent tag for each part of speech is shown in table 1.1. These POS tags have been deduced from Alpino's lexicon.

Past participles (psp) have been considered as a separate word class, as they show a lot of structural ambiguity in Dutch: past participles can systematically be used as verb forms as well as adjectives. By taking them as a separate class, the performance of the classifier on this specific task can be evaluated.

As the rest category contains only 13 examples, we will not pay any attention to it in the evaluation. To properly evaluate the other part of speech classes (such as Dutch adverbs), more training and test data is needed.

POS	n	most frequent tag
noun	226	noun(de,sg)
adjective	101	adjective(e)
past participle (psp)	53	adjective(no_e(adv))
verb	85	verb(hebben,pl,transitive)
rest	13	tmp_noun(tmp_de,sg)

Table 1.1: Most frequent tag for each part of speech

Table 1.2 shows the overall results of our algorithm (morphology & parse results), compared to the baselines, and compared to the use of a maxent classifier with only morphological information and a maxent classifier with only the information yielded by our parsing method. In tables 1.3-1.5, these results are split out for each part of speech.

Table 1.2 shows that our algorithm is able to reach a precision of 77.55% and a recall of 72.15%, yielding an f-score of 74.75%. The algorithm beats the naive baseline by 50%, and the more sophisticated baseline by about 35%. Our parsing method without the combination with morphology achieves already quite good results (yielding an f-measure of ± 71%). Combining this method with morphological information further improves these results, although recall decreases slightly.

Table 1.3 shows the results for the maxent classifier only trained on the morphological information of the word.

tributes of a certain tag is wrong, the whole tag is considered wrong. It may well be the case, however, that the algorithm missed out on only one certain subattribute of the tag, and found all other characteristics correctly.

[6]This is the score one might hope to attain by using an ordinary POS tagger.

	precision (%)	recall (%)	f-measure (%)
naive baseline	23.01	20.80	21.85
POS-based baseline	44.14	35.22	39.18
morphology	53.93	59.73	56.68
parse results	69.79	72.47	71.11
morphology & parse results	77.55	72.15	74.75

Table 1.2: Overall results

Morphology	precision (%)	recall (%)	f-measure (%)
noun	67.60	67.69	67.64
adjective	53.94	68.07	60.19
psp	44.11	56.94	49.71
verb	28.10	35.98	31.56
rest	25.00	23.08	24.00

Table 1.3: Evaluation of morphological information results

The results of table 1.3 seem to indicate that morphology is already quite a good indicator for noun and adjectives, but more complex syntactic features (such as the subcategorization frame of verbs) are evidently not found. In order to find these features, we need the results yielded by our universal parsing method. Those results are given in table 1.4.

Parse results	precision (%)	recall (%)	f-measure (%)
noun	88.53	84.56	86.50
adjective	66.37	82.43	73.53
psp	47.71	53.27	50.34
verb	41.96	43.33	42.63
rest	42.56	53.85	47.54

Table 1.4: Evaluation of parsing with universal tags

Table 1.4 shows that our parsing method scores better than the morphology classifier, with nouns reaching up to 86.5% and adjectives reaching up to 73.5%. This is an increase of respectively about 20% and 10% compared to morphology. Verbs, on the other hand, still have unsatisfactory results: the results of the past participle have not increased, and also the other verbs have low results (42.5% f-measure). The information conveyed by morphology is an important feature for verbs. In table 1.5, the results of the combination of both our parsing method and morphology are shown.

Morphology + parse results	precision (%)	recall (%)	f-measure (%)
noun	88.42	83.73	86.01
adjective	74.42	78.71	76.50
psp	71.98	54.37	61.95
verb	61.80	51.57	56.22
rest	38.46	26.92	31.67

Table 1.5: Evaluation of the combination of the universal parsing method with morphology

Combining the parsing method with morphology boosts the results for past participles and verbs with more than 10% (f-measure ± 62% and ± 56%). Morphology is indeed beneficiary for the verb results. Also, the adjective results increase slightly. The noun results, on the other hand, slightly decrease when combining with morphology, although this decrease does not outweigh the advantages for verb classification.

Common errors are found in the acquisition of adjective tags. The Alpino grammar contains a rather complicated adjective system. Different distinctions exist for adjectives that can be used predicatively, attributively, ... The algorithm is not always able to capture the correct subfeatures. Still, adjectives reach both precision and recall of about ± 75%.

Also, the acquisition of infrequent subcategorization frames, such as:

- ditransitive verbs
- verbs with prepositional complement
- verbs with locative complement
- verbs with sbar-complement

poses some problems. To classify verbs with these subcategorization frames properly, we probably need more data, and perhaps a small adaptation of our parsing method (see below).

5 Conclusion and Future Work

The evaluation of our lexical acquisition algorithm gives quite good results. The algorithm beats the naive baseline by 50%, and the POS-based baseline by 35%. 52% of the words get all (and only) the correct lexical type(s) (as specified in Alpino's lexicon). This approach shows that automatic lexical acquisition is certainly feasible. There is, however, still room for improvement, especially with regard to verbs.

Which brings us to some further research issues. First of all, it might be interesting to investigate the usefulness of a **cascading classifier**, i.e. a separate classifier for each part of speech. The results with regard to morphology (verb classification improves but noun classification gets worse) seem to indicate that this should indeed improve the overall result. Next, it might be useful to generalize over **word paradigms** when classifying words. Especially with regard to verbs, this generalization might improve the classification of subfeatures, such as the subcategorization frame of verbs.

It also remains to be investigated how to cope with **unknown multi-word expressions**. Coping with such expressions is likely to be a more difficult task, as the parsing method we use heavily relies on the context of words; if one of the context words is also an unknown word (as is the case with multi-word expressions), the method might have difficulties finding appropriate tags.

Next, it might be better to make use of the **top-n parses** (e.g. the 5 best parses) that are yielded by our parsing method. At the moment, only the best one is used, while the next best parses might contain quite useful information. This information should also be taken into account. This adaptation might give better results for infrequent subcategorization frames, as those frames might not make to the top-n parse due to their low a priori probability.

One last, important issue is to test our classification algorithm for unknown words on a **real test set of unknown words**. This will be done by testing Alpino's performance on a hand annotated test set that is not part of Alpino's treebank. Next, the lexical acquisition algorithm is applied to unknown words, the acquired tags are added to Alpino's lexicon, and the test set is parsed again, to see whether the results have improved. This way, it will become clear to what extent our classification algorithm might improve Alpino's coverage and accuracy.

Bibliography

Baldwin, T., E. M. Bender, D. Flickinger, et al. (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.

Barg, P. and M. Walther (1998). Processing Unknown Words in HPSG. In *Proceedings of the 17th International Conference on Computational Linguistics*, Volume 1, Montreal, Canada, pp. 91–95.

Bouma, G., G. van Noord, and R. Malouf (2001). Wide coverage computational analysis of Dutch. In W. Daelemans, K. Sima'an, J. Veenstra, and J. Zavrel (Eds.), *Computational Linguistics in the Netherlands 2000.*

Brent, M. R. (1993). From grammar to lexicon: unsupervised learning of lexical syntax. *Comput. Linguist. 19*(2), 243–262.

Erbach, G. (1990). Syntactic processing of unknown words. In P. Jorrand and V. Sgurev (Eds.), *Artificial Intelligence IV - methodology, systems,applications*, pp. 371–382. Amsterdam: North-Holland.

Fouvry, F. (2003, April 12–17). Lexicon acquisition with a large-coverage unification-based grammar. In *EACL 2003. 10th Conference of The European Chapter. Conference Companion*, Budapest, Hungary, pp. 87–90. ACL.

Le, Z. (2004). *Maximum Entropy Modeling Toolkit for Python and C++.* http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Prins, R. and G. van Noord (2001). Unsupervised POS-tagging Improves Parsing Accuracy and Parsing Efficiency. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT 2001)*, Beijing, China, pp. 154–165.

Schulte im Walde, S. (2002). Evaluating Verb Subcategorisation Frames learned by a German Statistical Grammar against Manual Definitions in the *Duden* Dictionary. In *Proceedings of the 10th EURALEX International Congress*, Copenhagen, Denmark, pp. 187–197.

van der Beek, L., G. Bouma, and G. van Noord (2002). Een brede computationele grammatica voor het Nederlands. *Nederlandse Taalkunde 7*(4), 353–374. in Dutch.

van Noord, G. (2004). Error Mining for Wide-Coverage Grammar Engineering. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, Barcelona, Spain, pp. 446–453.

van Noord, G. (2006). At Last Parsing Is Now Operational. In P. Mertens, C. Fairon, A. Dister, and P. Watrin (Eds.), *TALN06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, Leuven, pp. 20–42.