An Inverted Index for Storing and Retrieving Grammatical Dependencies

Michaela Atterer[*]**, Hinrich Schütze**[**]

[*]Institute for Linguistics
University of Potsdam
atterer@ling.uni-potsdam.de
[**] Institute for NLP
University of Stuttgart
hinrich@hotmail.com

Abstract

Web count statistics gathered from search engines have been widely used as a resource in a variety of NLP tasks. For some tasks, however, the information they exploit is not fine-grained enough. We propose an inverted index over grammatical relations as a fast and reliable resource to access more general and also more detailed frequency information. To build the index, we use a dependency parser to parse a large corpus. We extract binary dependency relations, such as *he-subj-say* (*he* is the subject of *say*) as index terms and construct the index using publicly available open-source indexing software. The unit we index over is the sentence. The index can be used to extract grammatical relations and frequency counts for these relations. The framework also provides the possibility to search for partial dependencies (say, the frequency of *he* occurring in subject position), words, strings and a combination of these. One possible application is the disambiguation of syntactic structures.

1. Introduction

Word count statistics retrieved from the world wide web have become a widely-used resource in a variety of NLP tasks, such as candidate selection for machine translation (Lapata and Keller, 2005), spelling correction (Lapata and Keller, 2004), and resolving attachment ambiguities (Volk, 2001). Sometimes, as in (Volk, 2001), raw word counts are found to be insufficient, because further linguistic information of some sort is paramount to increase performance.
In the case of PP-attachment for German, for instance, morphological information increases the performance of the system. The reason is that a more general linguistic representation than just co-occurrence of two word forms is required for this task. To disambiguate the PP attachment in the sentence

(1) The boy eats the cake with the spoon.

it is preferable to not only count occurrences of *eats* and *with the spoon* but also occurrences of *eat* and *with the spoon*. This is even more significant for languages with a richer inflectional morphology than English.
While being extremely useful, such extensions require further processing and further queries to the www and can thus become a problem for computational efficiency, especially when used as a component in an NLP system.
Other NLP applications like anaphora resolution require structural information, such as typical arguments for a verb. In languages with grammatical gender like German referring expression such as *er* (he) can refer to all entities with the corresponding gender. In the example below, it can refer to *death*, *owner*, *Volkswagen*, and *Sunday*.

(2) Nach dem [Tod] des [Besitzers] des [VW] am [Sonntag], ist klar, dass *er* erdrosselt wurde.

(3) Lit: After the [death] of-the [owner] of-the [Volkswagen] on [Sunday], it is clear, that *he* strangled was.

We cannot use surface string statistics from a search engine to resolve the anaphor *er*. For example, searching for *Besitzer wurde erdrosselt* ("owner was strangled") is problematic because another potential referent might appear next to the relevant verb. More generally, the actual referent need not appear near the verb at all:

(4) Der(The) Besitzer(owner) der(of-the-plural) Volkswagen(Volkswagen) wurde(was) erdrosselt(strangled).

(5) Der(The) Mörder(murderer) erdrosselte(strangeled) am(on) Sonntag(Sunday) den(the) Besitzer(owner).

In both 4 and 5 the referent (*the owner*) is not adjacent to *was*. In 4, *Volkswagen* is adjacent to *was strangled*. A search for surface strings might suggest that *Volkswagen* is a potential object of *strangle* in this case. To obtain a reliable answer, it is therefore necessary to parse the returned sentences and extract the actual object of the verb. This point is even more applicable to languages that do not have a fixed word order.
Post-processing of search engine results can become a problem for computational efficiency, for example if used in real-time applications like dialogue-systems. (Kilgarriff, 2007) argues against the use of search engines for computational linguistic research using similar arguments.
In this paper, we propose an inverted index over grammatical dependencies as an alternative to using the index of a web search engine. A dependency-index has the following advantages compared to a www search engine:

1. Greater linguistic generalization that makes possible the exact estimation of frequencies needed for many tasks. We can count, for instance, whether a noun *x* occurs as an object of a verb *y* even if the verb frequently occurs in the passive.

2. Fast accessibility of detailed linguistic knowledge. Not only are we able to count whether a noun occurs *near* a verb, but whether it actually occurs in the grammatical relation we are interested in. To achieve this with a www search engine, we would need a time-consuming online post-processing of our search results.

3. Greater reliability of the frequency counts. As is well known counts returned by search engines are unreliable (Kilgarriff, 2007) and we have to cope with limitations such as an upper limit on the number of queries a single machine is allowed to issue per day etc.

The remainder of this paper is organized as follows: Section 2 presents the resources we used to build our dependency index. Section 3 describes its functionality including the types of queries that are possible. Section 4 discusses possible applications, Section 5 describes related work, Section 6 discusses the limitations of the approach, and Section 7 concludes.

2. Resources for Building a Dependency Index

2.1. Corpus

As a corpus we used 80,000,000 words of the Reuters RCV1 corpus (Lewis et al., 2004). It contains newswire text, and about 3,820,057 relations like *obj(ect)* and *subj(ect)*. The triple "dog obj chase" or "cat subj chase" are examples of the relations obj and sub. The last two weeks of the Reuters corpus were set apart for future experiments.

2.2. Retrieving Dependencies – Minipar

To be able to index over grammatical dependencies, we annotated the corpus using Minipar (Lin, 1998). Minipar is a free partial dependency parser that outputs a dependency structure as shown in Figure 1.

```
(
E0 (() fin C * )
1 (The The Det 2 det (gov cat))
2 (cat cat N 3 s (gov chase))
3 (chases chase V E0 i (gov fin))
E2 (() cat N 3 subj (gov chase)
        (antecedent 2))
4 (the the Det 5 det (gov dog))
5 (dog dog N 3 obj (gov chase))
6 (.  . U * punc)
)
```

Figure 1: Minipar's (Lin, 1998) parse for the sentence "the cat chases the dog". The information that *cat* is the subject of *chase* can be extracted from line E2, and the information that *dog* is the object of *chase* from line 5.

2.3. Storing dependencies – Lucene Index

Given the syntactic analysis in Figure 1, we extract lemmata, such as *dog, cat,* binary dependency relations, such as *cat-subj-chase* and also partial dependency relations, such as *cat-subj,* which can be used to indicate the frequency of a lemma occurring with a specific function in a relation. These are used as index terms for a sentence. We can then retrieve all sentences that contain a lemma used with a particular function, or just the count of all those sentences. For example, we can retrieve the sentences that contain *cat* in the function of the subject (cf. Section 3.).

The terms are then stored in an inverted index (Witten et al., 1999) using Lucene (Lucene, 2006). The unit we index over is the sentence. Lucene allows a variable number of fields. Figure 2 shows an index containing three fields. One field stores the dependencies, one the surface string, and one field the number of the sentence. Note that for reasons of space efficiency, we omitted certain dependencies. For example we didn't index the surface subject s. Minipar's parse of the sentence *The dog is chased by the cat* contains the relations *dog-s-be* (*dog* is the surface subject of *be*) as well as *dog-obj-chase* (dog is the object of *chase*). The object relationship is useful for many tasks whereas the surface-subject relationship is arguably not usable in most NLP applications. Other dependencies can be omitted from the index to save space.

Also, we may not want to store the sentence itself (field 3 in 2) in cases where there is no need to do the type of phrase search that search engines support. It can, however, be useful to be able to combine surface string searches with grammatical relation searches.

2.4. Storage Requirements and Performance

The actual size of the index strongly depends on which dependency relations are actually indexed. In the applications described in (Atterer and Schütze, 2006b) and (Atterer and Schütze, 2006a) we used indexes from around 3G up to 18G for the Reuters data described above. These did not include fields with pure string data. The access time per query was in between approx. 3.8 and 4.8 seconds depending on the complexitiy of the queries and index entries.

3. Functionality

The design of the index as shown in Figure 2 enables us to perform queries of the following types:

- Queries for lemmas such as *cat*. We can access all sentences (both in their surface form and their parsed form) containing any morphological form of the base-form *cat*, and we can retrieve the exact number of such sentences.

- Grammatical relations such as *cat is the subject of chase* using a query like *cat-subj-chase*. Again we can access the sentences in the various forms stored as well as frequency counts.

- Partial dependencies such as *obj-dog*. Thus, we can count, how often *dog* occurs as the object of a verb.

- Approximations of complex dependency structures such as: Is *cat* the subject of *chase* and *dog* the object of *chase* in a sentence? We implement this by issuing queries containing boolean operators: *cat-subj-chase && chase-obj-dog* .

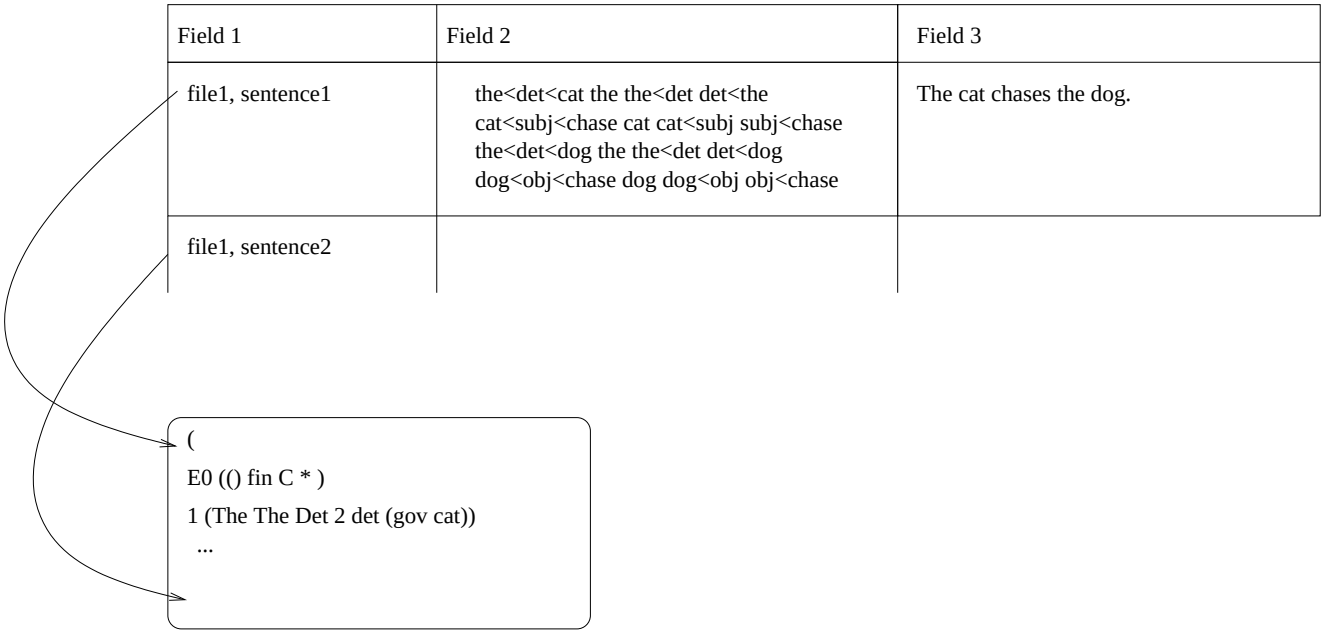

Figure 2: Schematic diagram of fields in the Lucene index, showing how dependencies can be used as search terms.

- When a third field as in Figure 2 is used, we can also query for substrings of the surface representation of the sentence, such as *cat chases*.
- Finally relating information of several fields is possible, too, such as querying for *cat chases* in combination with *cat-sub-chase*. This is of interest, when dependency information in combination with textual order is necessary. Processing French, for instance, one might be interested in dependency structures containing *une ancienne loi* (an old/obsolete law, not valid any more) or *une loi ancienne* (an old law, which has been valid for many years). The relevant dependency structure can be found in conjunction with the textual order in such cases.

All of these queries run much more efficiently on the dependency index than on a regular bags-of-word inverted index since no postprocessing of the sentences is necessary.

4. Applications

There are a number of potential applications for indexes over dependency relations. We have used the index for the tasks of relative clause (RC) attachment and prepositional phrase (PP) attachment (Atterer and Schütze, 2006b),(Atterer and Schütze, 2006a). Given the sentence

(6) The cat in London that chases the dog ...

and a corpus containing the two example sentences discussed above (*The dog chases the cat* and *The dog is chased by the cat*) we will find evidence for *cat* being the subject of *chase* and *dog* being the object of *chase* (2 sentences that contain both dependencies, as we generalize over passive and active use). But we will not find evidence for *London* being the subject of *chase*. Thus, we would be able to determine the correct attachment of the relative clause.
Considering Examples 7 and 8

(7) The house in London that Jack builds....

(8) The house in London that consists of....

an application searching the web for *house NEAR builds* cannot achieve results as good as one querying for *house* as the object of *build*.
In (Atterer and Schütze, 2006a), we showed that the attachment accuracy for relative clauses can be increased from 55.3% to 60.9% for *that*-relatives and from 73.1% to 78.4.% for *which*-relatives. The indexing framework presented in detail now, was a major component of this system.

5. Related Work

(Miyao et al., 2006) and (Ohta et al., 2006) use an HPSG parser to annotate bio-medical texts with predicate-argument structure. The sentences are stored in a structured database as opposed to an inverted index. The focus is on the bio-medical domain, i.e. relations of genes and proteins. The approach also allows for highly structured queries, and thus needs a more complicated indexing and search strategy. In contrast, our approach uses simple reformatting of text and off-the-shelf indexing technology that is mostly free and easily available. Our approach is thus an ideal alternative to using the web for researchers in linguistics and computational linguistics.
Alternative tools for searching for grammatical relations and structure in corpora are the sketch engine (Kilgarriff, 2007) and the linguist's search engine (Resnik and Elkiss, 2003). The sketch engine supports searching corpora for the grammatical behavior of words. It is possible to list

a word's objects together with frequencies, to search for words that behave similarly, and to show concordances. However, it is not possible to easily search for complex dependency structures and to do a combined search on grammatical structure and surface strings.

The linguist's search engine (Resnik and Elkiss, 2003) supports searching for arbitrary syntactic structures. Its main purpose is to help the "ordinary working linguist without considerable computer skills" to find examples of certain syntactic constructions. It is thus more complex than the relatively simple indexing procedure we present here, and has more overhead for the simpler queries we discuss in this paper.

The approach by (Bilotti et al., 2007) is most similar to the work we present here. The authors index a semantically parsed corpus and show that structured retrieval using this index can improve a Question Answering system. However, their main interest is weighted retrieval and they do not provide a simple way of querying for and counting dependencies.

6. Discussion

One of the limitations of the approach is that we can only approximate queries for complex dependency structures. We can retrieve the number of sentences where *cat* is the subject of *chase* and *dog* is the object of *chase*, but we cannot be sure that this query refers to the dependency structure in Figure 6., where the cat and the dog are part of the same event as opposed to two separate events of a chasing cat and a dog being chased:

Figure 3: Complex dependency structure with more than one dependency relation.

For example, the following sentence would also be retrieved by the query "cat subj chase AND dog obj chase":

(9) The cat chases the mouse, and Peter chases the dog.

7. Conclusion

We have proposed an inverted index data structure for dependency relations. It can be used as a fast, efficient and reliable NLP component. We have shown that it provides detailed linguistic information that cannot be obtained from web counts.

8. References

Michaela Atterer and Hinrich Schütze. (2006a). The effect of corpus size in combining supervised and unsupervised trainin g for disambiguation. In ACL Poster Proceedings, Sydney, Australia.

Michaela Atterer and Hinrich Schütze. (2006b). A lattice-based framework for enhancing statistical parsers with information from unlabeled corpora. In Proceedings of CoNLL, New York, USA.

Matthew W. Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. (2007). Structured retrieval for question answering. In SIGIR, pp. 351–358.

Adam Kilgarriff. (2007). Googleology is bad science. Computational Linguistics, 33(1), pp. 147–151.

Mirella Lapata and Frank Keller. (2004). The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In HLT-NAACL, pp. 121–128.

Mirella Lapata and Frank Keller. (2005). Web-based models for natural language processing. ACM Transactions on Speech and Language Processing, 2,pp. 1–31.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. (2004). RCV1: A new benchmark collection for text categorization research. The Journal of Machine Learning Research, 5, pp. 361–397.

Dekang Lin. (1998). Dependency-based evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain.

Lucene. (2006). http://lucene.apache.org.

Yusuke Miyao, Tomoko Ohta, Katsuya Masuda, Yoshimasa Tsuruoka, Kazuhiro Yoshida, Takashi Ninomiya, and Jun'ichi Tsujii. (2006). Semantic retrieval for the accurate identification of relational concepts in massive textbases. In Proceedings of COLING-ACL 200, Sydney, Australia.

Tomoko Ohta, Yusuke Miyao, Takashi Ninomiya, Yoshimasa Tsuruoka, Akane Yakushiji, Katsuya Masuda, Jumpei Takeuchi, Kazuhiro Yoshida, Tadayoshi Hara, Jin-Dong Kim, Yuka Tateisi, and Jun'ichi Tsujii. (2006). An intelligent search engine and GUI-based efficient MEDLINE search tool based on deep syntactic parsing. In Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, Sydney, Australia.

Philip Resnik and Aaron Elkiss. (2003). The linguist's search engine: Getting started guide. Technical Report LAMP-TR-108/CS-TR-4541/UMIACS-TR-2003-109, University of Maryland, College Park.

Martin Volk. (2001). Exploiting the WWW as a corpus to resolve PP attachment ambiguities. In Proceedings of Corpus Linguistics 2001.

Ian H. Witten, Alistair Moffat, and Timothy C. Bell. (1999). Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufman.