LiSa – morphological analysis for information retrieval

Hans Hjelm
CL Group, Department of Linguistics
Stockholm University
SE-106 91 Stockholm, Sweden
hans.hjelm@ling.su.se

Christoph Schwarz
Intrafind Software AG
Lortzingstraße 9
DE-81241 Munich, Germany
christoph.schwarz@intrafind.de

Abstract

This paper presents LiSa, a system for morphological analysis, designed to meet the needs of the Information Retrieval (IR) community. LiSa is an acronym for *Linguistic and Statistical Analysis*. The system is lexicon- and rule based and developed in Java. It performs lemmatization, part of speech categorization, decompounding and compound disambiguation for German, Spanish, French and English, with the other major European languages under development. The lessons learned when developing the rules for disambiguation of German compounds are also applicable to other compounding languages, such as the Nordic languages. Since compounding is much more common and far more complex in German than in the other languages currently handled by LiSa, this paper will deal mainly with German.

A comparative evaluation of LiSa with the GERTWOL system[1], combined with a filter for disambiguation developed by Volk (Volk, 1999) has been performed. (The combination of GERTWOL and filter will from here on be referred to as *Filtered GERTWOL*.) The focus of the evaluation has been to measure how suitable the respective systems are for query processing and for building indices for IR-systems. Special attention has been paid to their abilities to select the correct analysis of compounds.

LiSa is developed by Intrafind Software AG, on whose homepage an online demo of LiSa can be found[2]. It is used in Intrafind's *iFinder* and also exists as an add-on to the open source free text indexing tool *Lucene*[3].

1 Introduction

(Sproat, 1992), p. 7, states that

Lemmatization is normally not an end in itself, but is useful for other applications, such as document retrieval (...) or indexing.

The cornerstone of almost any IR-system is the index; a table structure, showing which words appear in which documents, possibly also in which order and how often. The following sections describe how lemmatization, decompounding and compound disambiguation is helpful when constructing and searching the index. The usefulness of lemmatization has been questioned for English (Harman, 1991), but is asserted for other languages (Hull, 1996). (Note that these papers deal with *stemming* rather than lemmatization. Stemming is a more aggressive approach, reducing even words from different part of speech (POS) categories to common stems.) For compounding languages, research has shown that decompounding is a worthwhile effort (Braschler and Ripplinger, 2004). It should be noted that it is important to use the same methods of analysis for query processing as for constructing the index, e.g., if one uses lemmatization during query processing, one must use lemmatization also during indexing.

[1] http://www.lingsoft.fi/doc/gertwol/

[2] http://www.intrafind.org

[3] http://jakarta.apache.org/lucene

S. Werner (ed.), *Proceedings of the 15th NODALIDA conference, Joensuu 2005*, Ling@JoY 1, 2006, pp. 65–70
ISBN 952-458-771-8, ISSN 1796-1114. http://ling.joensuu.fi/lingjoy/

1.1 Lemmatization

Using a lemmatizer when constructing the index brings the main advantage of an increase in recall. Searching for “Bücher” (books), the system will also find documents containing “Buch” (book) or any of the other inflectional forms of that same word. Note that the usual trade-off between precision and recall does not necessarily apply here. In fact, Braschler and Ripplinger (2004) argue that precision might *increase* along with recall.

1.2 Decompounding

This is the process of splitting a compound word into its parts. E.g., the word “Bücherregale” (book shelves) would be split into the parts “Bücher” (books) and “Regale” (shelves). Usually this process is combined with lemmatization, to give the citation form[4] of the constituents: “Buch” (book) and “Regal” (shelf). These constituents are then added to the index, along with the citation form of the entire compound (here: “Bücherregal” (book shelf)). The purpose of decompounding is mainly to improve recall.

1.3 Compound disambiguation

Often more than one reading is possible for a complex word. The word “Kulturteilen” has (at least) four possible readings:

```
1. "Kultur (noun) + teilen (verb)" (to
share culture)
2. "Kultur (noun) + teilen (noun)"
(the culture section of a newspaper)
3. "Kult (noun) + urteilen (verb)" (to
judge a cult)
4. "Kult (noun) + urteilen (noun)"
(cult judgments)
```

Although all four readings are possible, the second reading is markedly more probable than the others. Compound disambiguation is the process of finding this most probable reading and is used within IR to increase precision in a system. This paper focuses mainly on this last task, since it is by far the most challenging one.

2 Related work

There exist a great number of systems for morphological analysis, both commercial and in the research community. It is not the purpose of this paper to give an exhaustive overview of the existing systems in the field of computational morphology. This section singles out relatively recently developed systems for the Nordic languages. Of course, GERTWOL (Haapalainen and Majorin, 1994), the system we use as a reference for our tests in this paper, is one of the more well-known for the German language. It is described in section 6.

A statistically based system Sjöberg and Kann (2004) describe a system for morphological analysis, decompounding and compound disambiguation for the Swedish language. They use a variety of measures, including looking at the number of components in the analysis, analyzing the frequencies of words and POS-categories in the context and the POS-categories of the components. The best results, an accuracy of 94% for ambiguous compounds, are reached by using a hybrid system, where the most successful statistical measures are combined with some ad hoc rules.

A rule-based system For Norwegian, Bondi Johannesen and Hauglin (1996) report of a system for decompounding and compound disambiguation to be used in a system for morphosyntactic tagging. Their approach resembles the one used in LiSa (see description in section 4), in that both systems have a hierarchy of rules that are used for the disambiguation. They report of an accuracy of 97.6%, where mistakes depending on missing lexicon entries were not counted as errors.

3 Aspects of analyzing for IR

Here we look at some further considerations that are important for IR, specifically, when performing morphological analysis.

3.1 Depth of analysis

Many systems use a strategy where a large list of inflected forms are kept in a dictionary, along with their citation forms. If the system finds the word the user is looking for, e.g., “Kulturteilen”, the base form is returned to the user (“Kulturteil (noun)”). In such systems, decompounding is only used as a fallback option when the lexicon fails to deliver the base form. This will give the user the correct base form, but she or he will not be able to use the constituents for indexing. This is an example where the analysis is too shallow.

On the other hand, a very deep analysis is also of little use for indexing. E.g.,

[4] the word as it would appear in a dictionary entry

it would be possible to identify the following morphemes in the word “Destabilisierungsvorgang” (procedure for destabilization): “De|stabilisier|ung|s|vor|gang”. Including all these morphemes in the index would clutter it and decrease the precision and performance of the system. Here, then, the analysis is too deep - what we need is simply the main constituents of the compound: “Destabilisierungs|vorgang” (the respective lemmas being “Destabilisierung” and “Vorgang”).

3.2 Category system

When building an index, the detailed POS-category of a word is of less importance. Since terms will be added to the index in citation form, and this form always has the same grammatical features for each POS-category, it would be enough to differentiate between the basic POS-categories, like nouns, verbs and adjectives. E.g., nouns will always be added to the index in the singular nominative form; therefore no case or number information appears in the index, only the basic POS-category is stated. However, when performing compound disambiguation, more detailed information can sometimes be useful (see 3.3). The category system used in LiSa is tailored towards this kind of analysis and disambiguation (two levels of detail are available during the analysis). The category system has also been developed with the detection of multi-word units in mind, but this paper will not explicitly deal with that topic.

3.3 Treatment of upper/lower case

Making use of information on capitalization can sometimes provide assistance in disambiguation. The obvious example for German is that nouns are capitalized in running text. If we therefore have more than one competing readings, all noun readings can be excluded if the word is not capitalized. E.g., upper case “Wein” can be both the noun (wine) and the verb (cry, second person imperative), since verbs and adjectives are also capitalized at the beginning of a sentence. Lower case “wein” though, can only be the verb reading. One also needs to take two other types of texts into consideration:

- **Orthographically non-standard texts**. This group includes, but is not limited to, e-mails (which are often all lower case) and certain web pages (containing mainly lists or tables). Since these texts do not comply with standard capitalization rules, it is not possible to use capitalization information for disambiguation here.
- **Citation form**. Here the words appear as they would in a dictionary. This form actually allows for more precise disambiguation, since one also can exclude any upper case verb or adjective readings.

As can be seen in the previous example with “wein”, capitalization can be used for disambiguating not only compounds, but simplex words as well. Similarly, for English, one can use the knowledge that verbs in second person never appear sentence initially, to rule out the verbal reading of, e.g., “Uses” (capitalized), leaving only the nominal reading.

3.4 Analysis of special characters (umlauts)

Some languages with so called special characters have a representation of these characters that is possible to write using an arbitrary keyboard for the Latin alphabet. This is the case for German, but also for Swedish and other Nordic languages. For German, the mapping looks like this:

```
ä -> ae
ö -> oe
ü -> ue
ß -> ss
```

Taking this possibility into consideration when indexing can increase the recall of a system, but also adds complexity, since these letter combinations also occur naturally in the language, especially at word boundaries in compounds. E.g., for “Bau|experte” (building expert), analysis will fail if it is first processed to “Baüxperte”. Conversely, we will not be able to analyze “Drueck|experte” (printing expert), unless we first process it to “Drück|experte”.

4 LiSa word analysis

This section describes the approaches taken in LiSa towards solving the issues discussed in this paper.

4.1 Lemmatization in LiSa

The backbone of the analysis in LiSa is the lexicon, stored in a letter tree format for fast and space efficient access (for running text, LiSa processes over 150.000 German words per second on a Pentium 4 machine with 512 MB RAM). The lexicon is a full form lexicon with mostly non-compound words, i.e., all possible inflectional forms of a word are stored, along with their respective lemmas and POS-categories. The LiSa lexicon distinguishes between simplex words, capable of appearing by themselves or as heads of compounds and words which can only appear inside compounds (and never as compound heads).

4.2 Decompounding in LiSa

If no complete match for a word is found in the lexicon, LiSa assumes we are dealing with a compound and produces all possible readings of the word, based on what the lexicon allows, using combinatorial rules. It is also possible to code compounds directly in the lexicon, which can be worthwhile for compounds which do not adhere to standard analysis patterns. LiSa gives the lemmas of the compound constituents, in addition to the lemma of the entire compound, also when the compound is coded in the lexicon.

4.3 Compound disambiguation in LiSa

During the decompounding step, it is frequently the case that a number of possible readings are produced. LiSa possesses a rule machinery with filtering rules for each language, some general and some language specific. The rules are chained together, each rule possibly reducing the number of possible readings and passing the rest along to the next rule in the chain, until, ideally, only one reading is left. The rules with the greatest coverage appear at the top of the rule hierarchy, for the sake of efficiency. Since each rule reduces the set of possible readings, the ordering of the rules is also important for producing the correct results. Some rules depend on other rules having been applied previously to function correctly.

In most cases, the goal of compound disambiguation is to be left with only one reading. In some rare cases, looking at a word in isolation, it is not possible to determine which reading is the more probable. E.g., the word “Nordpolen” has three almost equally probable readings:

```
"Norden + Pol" (North Pole)
"Norden + Pole" (Person from the north
of Poland)
"Norden + Polen" (Northern Poland)
```

The idea behind LiSa is to extract the most precise information possible on a word level. Once this has been done, tools working on a higher level (e.g. sentence level) can make use of this information and have a higher chance of succeeding. In the majority of cases, though, the information available on a word level is sufficient for performing the disambiguation.

The most basic filtering rule consists in choosing the reading with the smallest number of constituents. This rule would be effective in resolving the following ambiguity:

```
"Wohnungs|einrichtung" (Room
furnishing)
"Wohnungs|ein|richtung" (One direction
of a room)
```

Although the second reading is nonsensical, its constituents are all legitimate words and the reading has to be ruled out.

Another basic rule is to choose the reading with the longest right-most constituent. Here is an example for which this rule is applicable (the first alternative being the correct one):

```
"Erb|information" (inheritance
information)
"Erbin|formation" (heiress formation)
```

For ruling out particularly unlikely readings, some words get a marking in the dictionary, indicating that readings containing this word as a constituent should be disfavored when other readings are available. One example where this is put to use is the following (again, the first reading is the preferred one):

```
"Himmels|achse" (axis of heaven)
"Himmel|sachse" (heaven Saxonian)
```

Here, “sachse” has been marked as undesired in the dictionary, and hence the reading “Himmel|sachse” is filtered out, although it would be the preferred reading according to the rule concerning longest right-most constituent. A similar concept is described in (Volk, 1999). Words for which it is necessary to side-step this behavior, e.g., “Kursachse” (a kind of Saxonian), can be coded explicitly as “Kur|sachse” in the LiSa lexicon.

There are many other filtering rules implemented for German in LiSa. Some of them are specific to German, but most of them will carry over to other compounding languages, like Dutch and the Nordic languages.

5 Evaluation

To measure the quality of the analysis in general and the disambiguation in particular, a contrastive evaluation was carried out. The data for the evaluation comes from the articles appearing in the Swiss newspaper *Neue Zürcher Zeitung* in April 1994. From all articles in the data collection (near 3000 articles, totally about 1.8 million words), the longest words containing only alphanumerical characters were selected (306 word types). This collection is here referred to as `nzz_long` and is meant to present the most challenging task for compound disambiguation, the idea being that a longer word will contain more possibilities for ambiguity than a shorter word.

5.1 nzz_long

The results for the `nzz_long` test set have been processed for LiSa and Filtered GERTWOL. Although originally consisting of 306 types, four of these turned out to be typos or words written in a non-standard way, making the actual number of test words 302.

It should be noted that the errors reported are clear-cut errors - for less clear cases, we have adopted a more lenient approach. For example, one might argue whether “Infrastruktur” (infrastructure) should be analyzed as “Infra|struktur” or left as it is. Rather than deciding on a “correct” way in these murky cases, we have chosen to give both interpretations the benefit of the doubt. The results reflect the state of the systems as of April 2005.

In addition to the brief evaluation described here, it would be interesting to perform an application based evaluation. Using the test data from TREC[5] would provide valuable information, since one would be evaluating the effectiveness with regards to IR directly, which is what we are mainly interested in here.

6 Discussion - the systems contrasted

Here are some of the main issues that set LiSa apart from Filtered GERTWOL in terms of their aptness in an IR-environment.

6.1 Depth of analysis

The analysis produced by Filtered GERTWOL is more fine-grained than the one Lisa produces. This might at first seem like a pleasant problem - the superfluous information can simply be ignored. However, this is not as simple as it might at first sound. E.g., this is the analysis given for the word “Destabilisierungsvorgang”:

```
"*de|stabil~is~ier~ung\s#vor|gang"
```

Here, the strategy would be to split the word at the #-sign, get rid of the bounding morpheme after the \-character and use “destabilisierung” and “vorgang” for indexing. However, for the word “aufschreiben” (write down), we get the following analysis:

```
"auf|schreib~en"
```

There is no #-sign splitting the word in this case. Still, one would have liked to add at least “schreiben” (write) to the index. It is not a clear-cut case, which constituents to add to the index and which not. In LiSa, this problem does not arise, since the constituents delivered are always the base forms to be used for indexing.

6.2 Category system

The POS-category system used in Filtered GERTWOL is again more detailed than the one used in LiSa. Just as described in the previous section, this actually confuses rather than helps - a user of the system will have to post-process the output to get rid of unwanted duplicates. For the word “Bücherregale”, Filtered GERTWOL produces the following readings, which are identical except for their POS-categories:

```
"*büch\er#regal" S NEUTR PL NOM
"*büch\er#regal" S NEUTR PL AKK
"*büch\er#regal" S NEUTR PL GEN
"*büch\er#regal" S NEUTR SELTEN SG DAT
```

Again, LiSa produces a single output for this case, giving exactly the information needed for indexing or query analysis.

6.3 Modularity

LiSa can easily be used as a module in a bigger software system, since it is equipped with a well defined API and since it is written in Java. GERTWOL, or especially Filtered GERTWOL, does not lend itself to system integration in the same way as LiSa does.

6.4 Further differences

The differences listed in the following section are not less substantial than the ones described in sections 6.1 to 6.3. However, they have all been discussed previously in section 3 and are

[5] http://trec.nist.gov

	No analysis	Incorrect analysis	Ambiguous analyses
LiSa	1.0% (3)	0.3% (1)	0% (0)
GERTWOL	2.0% (6)	0.7% (2)	4.0% (12)

Table 1: *Contrastive evaluation, LiSa and Filtered GERTWOL. The first column counts words for which no analysis was found. The second column counts words for which one or more analyses were found, but none of them were correct. The final column counts words for which more than one analysis was given (only one analysis is correct for each word).*

therefore given briefer descriptions in the following.

- **Lemmas of compound constituents** Turning again to the example of "Bücherregale" from section 6.2, one sees that GERTWOL splits the noun, but the first constituent is still presented in its text form ("Bücher"). This seems counterproductive; we would like to get the base form of the entire compound ("Bücherregal") but also the base forms of the constituents ("Buch" and "Regal"), which is precisely the analysis given by LiSa.
- **Special characters treatment** Another difference between the two systems, is their ability to deal with special characters (see section 3.4 for a description of the problem). This type of analysis is especially useful when analyzing queries, but certain types of texts (e.g. e-mails) also use this kind of conventions.
- **Filtering** In addition to the issues raised previously, with regards to filtering, the Filtered GERTWOL system relies on capitalization complying with the citation form of words, which will produce filtering errors when analyzing running text.
- **Capitalization issues** Filtered GERTWOL does not allow for treating texts differently, depending on their origin or the type of text, in the way LiSa does.

7 Conclusions

Perhaps more than the numbers presented in section 5, the differences described in the previous section point to the usefulness of LiSa in the IR setting. Considering also that LiSa is able to process the text efficiently, both in terms of time and resources, and its availability as a plug-in to the widely distributed Lucene engine, we believe LiSa will prove to be a valuable asset for many IR applications.

References

Martin Braschler and Bärbel Ripplinger. 2004. How effective is stemming and decompounding for German text retrieval? *Information Retrieval*, 7(3–4):291–316.

Mariikka Haapalainen and Ari Majorin. 1994. Gertwol: ein System zur automatischen Wortformerkennung deutscher Wörter. Technical report, Lingsoft, Inc., September.

Donna Harman. 1991. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7–15.

David A. Hull. 1996. Stemming algorithms - a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84.

Janne Bondi Johannesen and Helge Hauglin. 1996. An automatic analysis of Norwegian compounds. In T. Haukioja, editor, *Papers from the 16th Scandinavian Conference of Linguistics*, pages 209–220, Turku/Åbo, Finland.

Jonas Sjöbergh and Viggo Kann. 2004. Finding the correct interpretation of Swedish compounds a statistical approach. In *Proceedings of LREC-2004*, pages 899–902, Lisbon, Portugal.

Richard Sproat. 1992. *Morphology and Computation*. The MIT Press, Cambridge, Mass, USA.

Martin Volk. 1999. Choosing the right lemma when analysing German nouns. In Jost Gippert and Peter Olivier, editors, *Multilinguale Corpora: Codierung, Strukturierung, Analyse*. Gesellschaft für Linguistische DatenVerarbeitung, Enigma Corporation.