Analyzing German Noun Compounds using a Web-Scale Dataset – Task description

UIMA Software Project WS 2010/2011
Jens Haase

1 Introduction

Noun-compounding in the German language is a powerful process to build new words with the combination of two or more existing words. *Blumensträuße* (flower bouquet) for example contains the two words *Blumen* (flower) and *Sträuße* (bouquet).

In Unstructured Information Management this can be a handicap for tools like Machine Translation, Speech Recognition or Information Retrieval. Imagine you search for the word *Lackschicht* (paint layer), the expected result will be all documents containing the word *Lackschicht* but also the words *Schicht aus Lack* (layer of paint). To find a expected result the search engine must split all compounds nouns in single words (*Lackschicht -> Lack schicht*).

2 Problem definition

In the German language a compounds can be formed by combining nouns, verbs and adjectives. Also compound words can combined to a new compound word. The word *Donaudampfschifffahrtskapitänsmütze* (a cap for captains of steamboats driven on the river Donau), contains the words *Donaudampfschiff* (steamboat on the river Donau) and *Kapitänsmütze* (the cap of the captain). Each word can be split again. The final decompounds words are *Donau*, *Dampf*, *Schiff*, *Fahrt*, *Kapitän* and *Mütze*.

Another problem in noun decompounds are linking morphemes. These morphemes are allowed between two words. The linking morpheme in *Kaptiänsmütze* is the the *s* between *Kapitän* and *Mütze* (*Kapitän(s)+mütze*). Also the word *Orangensaft* (orange juice) has a *n* as linking morpheme (*Orange(n)+saft)*. But in this case the word *Orangen* is a valid word in the German language. It is the plural of the word *Orange*. But the word *Kapitäns* in previous example is not a valid word.

At least not all possible splits have the same meaning. The word *Tagesration* (recommended daily amount), contains the word *Tag* (day), *Rat* (advice) and *Ion* (ion) but also the word *Ration* (ration). Splitting the the word in three parts is wrong, because the chemical ion has nothing to do with the word *Tagesration*. The correct decompounding here will be *Tag(es)+ration*.

3 Task

Decompounding of a word is mostly done in three steps [ea08]:

1. Calculate every possible way of splitting a word in one or more parts
2. Score those parts according to some weighting function
3. Take the highest-scoring decomposition. If it contains one part, it means that the word in not a compound.

At first we have to split a word in all possible parts. For that we need a dictionary with all existing words. It is recommend to use a special dictionary for the analyzing corpus. This can simple be done in extracting all tokens of a document collection. A list of linking morphemes is also needed.

In the next step we need a weighting function for each decompound possibility. The can be for example a simple count frequency of every word. The words with more frequency get a higher weight than other. At the end we can rank the weights. The decomposition with the highest weight wins.

At the end we want to have UIMA component that can be used in other projects. It should able to word with special dictionaries and linking morphemes. Optional different weighting function can be chosen.

4 Tools

4.1 Dictionary

As mentioned early it is recommend to build a dictionary for a special corpus, but in this case we do not have a special corpus. Instead we want to use a *every day* dictionary, which are used for spell correction. ASpell and ispell are open source spell checker with dictionaries for nearly every language. Also hunspell is a modern spell checker, used in OpenOffice, Mozilla Firefox and Google Chrome. The dictionaries of aspell and hunspell are mostly the same.

4.2 Google Web1T

Google Web1T corpus contains over 130 billion tokens of the German language, distributed over n-grams ($n \epsilon [1,5]$). The corpus can be used for the weighting function. The unigrams can also be used as dictionary.

To search and access the data I will use *jWeb1T* (`http://hlt.fbk.eu/en/technology/jWeb1t`). This is an open source tool for efficiently searching on the Google Web1T corpus.

5 Roadmap

Week	**Goals**
08.11 - 14.11	get familiar with the project; choosing dictionary
15.11 - 21.11	access to dictionary
22.11 - 28.11	access Google Web1T; splitting words
29.11 - 05.06	splitting words
06.12 - 12.12	splitting words
13.12 - 19.12	evaluation and testing
20.12 - 26.12	weighting function (Christmas)
27.12 - 02.01	weighting function (Christmas, new years eve)
03.01 - 09.01	weighting function
10.01 - 16.01	no time (vacation)
17.01 - 23.01	evaluation and testing
24.01 - 30.01	UIMA Component
31.01 - 06.02	project cleanup

5.1 Next weeks

In the next week I write a Component that can access the directory. The dictionary is either the hunspell dictionary or the unigrams in the Web1T dataset. I also want to create a list of compound words with the best split. This can help me to see a lot of variants for the implementation. The list can also be used for testing and evaluation.

References

[ea08] Enrique Alfonseca et al. German decompounding in a difficult corpus. In *Computational Linguistics and Intelligent Text Processing*, 2008.