Analyzing German Noun Compounds using a Web-Scale Dataset – Final presentation

UIMA Software Project WS 2010/2011

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Motivation
- Problem definition
- Splitting algorithm
- Ranking algorithm
- Lessons learned / Conclusion

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

Let's start with an experiment

- Take a search engine
- Search for: *Blumensträuße* (flower bouquet)
- As result you will receive documents with the word *Blumensträuße*
- You will also see document with the words *Blumen* (flower) and *Sträuße*

Result

- The search engine is intelligent and knows that *Blumensträuße* is a noun-compound

Problem definition (1) - Noun Compounding

TECHNISCHE UNIVERSITÄT DARMSTADT

Definition

A noun compound word is the combination of one or more individual words to a new word

Example

Blumensträuße -> Blumen + Sträuße

- Compounds are formed with nouns, verbs and adjectives.
- Compound words can be compound with other
- Linking morphemes are added between words: *Tag(es)+ration*
- Different context for different splits: *Tag(es)+ration* vs. *Tag(es)+rat+ion*

Problem definition (2) - Noun Decompounding

TECHNISCHE UNIVERSITÄT DARMSTADT

How to split a word? [ea08, Alfonseca et al.]

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.

- The algorithm for task 1 and task 2 two are nearly independent
- But results of task 2 can never be better the result of task 1
- Task 3 is the combination of task 1 and task 2

Splitting Algorithm (1) - Requirements

TECHNISCHE UNIVERSITÄT DARMSTADT

Problem 1

How do we know when a word starts or ends?

Solution

- Use a dictionary
- The following algorithm use the IGerman98 dictionary. This is part of most spell-checkers today.

Problem 2

How to evaluate?

Solution

- Use a Marek' corpus [Mar06, Marek] (around 160,000 examples)
- If the correct split in the list of all possible splits we have a correct result

Splitting Algorithm (2) - Left to right algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

- Walk from left to right through the word an check if left part is a correct word
- Right part can be an *unword*
- Result of real implementation returns a tree -> can be visualized

```
function split(word)
  result = List()
  for (i = 0..word.len)
    left = word[0..i+1]
    right = word[i+1..word.len]

    if (Dictionary.contains(left)
        and (right.len > 2 or right.len == 0))
      result += (left, right)

  return result
```

Splitting Algorithm (3) - Data driven algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

- Uses “statistics” of the dictionary
- With statistics I mean to put the dictionary in a trie.
- Example extracted from [ea00, Martha Larson et al.]

f	r	i	e	d	e	n	s	p	o	l	i	t	i	k
-	-	39	29	29	25	24	23	3	1	1	1	1	1	1
1	1	1	1	1	2	7	37	88	89	89	92	99	-	-

f	r	i	e	d	e	n	s	p	o	l	i	t	i	k
-	-	-	10	0	4	1	1	20	2	0	0	0	0	0
0	0	0	0	0	5	30	51	1	0	3	7	-	-	-

f	r	i	e	d	e	n	s	p	o	l	i	t	i	k
-	-	-	-		*			*						
							*				-	-	-	-

Splitting Algorithm (4) - Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

Algorithm	Correct with morphemes	Correct without morphemes
Left to right	0.813	0.888
Data driven	0.16	0.41

Legend

- Correct *without* morphemes means that only the splits are at the correct position, but the morphemes are not set correctly
- Correct *with* morphemes means that the splits are at the correct position and also the morphemes are set correctly

Result

- Left to right algorithm works a lot better then the other.
- Following always use the left to right algorithm

Ranking algorithm (1) - Requirements

TECHNISCHE UNIVERSITÄT DARMSTADT

Problem 1

To rank we need knowledge about the words

Solution

The Google Web1T corpus has frequency information about a lot of n-grams.

Problem 2

How to search the n-grams?

Solution

A lucene index was created. It has a size of 12.8 GB and is very slow on a normal hard disk

Ranking algorithm (2) - Frequency based [ea08, Alfonseca et al.]

$$F_s = \prod_{s_i \in S} freq(s_i))^{\frac{1}{|S|}} \tag{1}$$

- *S*: The split with split elements in it
- *freq*: Searches for n-grams with the given words and returns the frequency value from the corpus.

- For each split F_s is calculated.
- The split with the highest F_s wins

Ranking algorithm (3) - Probability based [ea08, Alfonseca et al.]

TECHNISCHE UNIVERSITÄT DARMSTADT

$$P_s = \sum_{s_i \in S} -log(\frac{freq(s_i)}{F}) \tag{2}$$

- *S*: The split with split elements in it
- *freq*: Searches for n-grams with the given words and returns the frequency value from the corpus.
- *F*: The total amount of frequency values in the corpus (add all values)

- For each split P_s is calculated.
- The split with the lowest P_s wins

Ranking algorithm (4) - Mutual information based [ea08, Alfonseca et al.]

TECHNISCHE UNIVERSITÄT DARMSTADT

$$M(w_1, w_2) = log_2\left(\frac{F \times freq(w_1, w_2)}{freq(w_1) \times freq(w_2)}\right) \qquad (3)$$

- *S*: The split with split elements in it
- *freq*: Searches for n-grams with the given words and returns the frequency value from the corpus.
- *F*: The total amount of frequency values in the corpus (add all values)

- *M* will be calculated for neighbor pairs in the split
- The value of a split is the average of all *M*
- The split with the highest averaged *M* wins

Ranking algorithm (5) - Evaluation

Algorithm	Correct tree	Correct@1	Correct@2	Correct@3
Frequency	**0.523 (0.720)**	0.508 (0.703)	0.665 (0.785)	0.726 (0.829)
Probability	0.182 (0.252)	0.184 (0.256)	0.510 (0.658)	0.628 (0.743)
MI	0.295 (0.442)	0.369 (0.542)	0.516 (0.666)	0.587 (0.737)

- *Correct tree*: Correct result when ranking on a tree (subset of a list)
- *Correct@1*: The first of the ranked list is correct
- *Correct@1*: The first or second of the ranked list is correct
- *Correct@1*: The first, second or third of the ranked list is correct
- Values in brackets are correct without morphemes

Lessons Learned / Conclusion

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- Lucene indexes on normal disks are very SLOW
- I need a better machine ;)

- Weekly documentation help to write final report, and to think about next steps

- Simplest algorithm returned best results

End

Questions

Ask now, or later.

More information

Code, documentation and slides are available on github:

`https://github.com/jenshaase/noun-decompounds`

References

TECHNISCHE UNIVERSITÄT DARMSTADT

- Martha Larson et al.
  Compound splitting and lexical unit recombination for improved performance of a speech recognition system for german parliamentary speeches.
  In *Proceedings ICSLP 2000: Sixth International Conference on Spoken Language Processing*, 2000.
- Enrique Alfonseca et al.
  German decompounding in a difficult corpus.
  In *Computational Linguistics and Intelligent Text Processing*, 2008.
- Torsten Marek.
  Analysis of german compounds using weighted finite state transducers.
  Master's thesis, Eberhard-Karls-Universität Tübingen, 2006.