Statistical Text Segmentation with Partial Structure Analysis

Felix Golcher
Humboldt-Universität zu Berlin
felix.golcher@hu-berlin.de

Abstract

This paper investigates a method for segmenting unannotated natural language text into morphemes or bigger linguistic units and for partially grouping them hierarchically. The approach is unsupervised and language independent. It is purely statistic and based on the frequency counts for all substrings in a text. The underlying model is very simple and has no free parameters.

1 Introduction

The proposed method for unsupervised text segmentation and structure inference is opposed to most approaches to natural language processing as it addresses the tasks of text segmentation, morphological decomposition, multiword unit detection, and compound analysis as a whole. Usually these tasks are handled separately in subsequent work stages. See Manning and Schütze (1999) for general introductions to these fields.

The linguistic prerequisites which enter the model are minimal. This makes its implementation an investigation on how far one can get with an approach of unsupervised language structure learning which solely relies on surface statistics.

The results show clearly that the task of text segmentation can be solved fairly well along this line, while treelike grammatical structures cannot be covered equally well. Success and failure both offer insight into the relation between statistics and language structure.

2 Related Work

Nearly all statistical approaches to unsupervised morpheme detection are either related to the work of Harris (1955) and Hafer and Weiss (1974) or to Goldsmith (2001).

The first approach takes a substring of a text and counts the number of possible next characters as occurring in some training corpus. For example this number is 3 for *instrument*[1]: possible continuations are *a*, *s*, and _. The text is segmented at those points, where this *successor variety* has a local maximum. Bordag (2005), Dang and Choudri (2005) and Hammarström (2006) fall into this category.

The second approach tries to find a description of the training corpus whose length is minimal. The length of the description is the sum of the lengths of its two parts, the morpheme inventory and the model used to combine these morphemes into words. A rather successful example of such an approach is Creutz and Lagus (2005).

Approaches of both kind usually operate on pretokenized text or even on word lists and aim at inferring word morphology. In contrast, the proposed method for analysing natural language texts starts from the raw character sequence and aims at finding linguistic units not confined to word boundaries. This is both more interesting and more promising, since obviously nearly all information carried by a text is destroyed when the text is compressed into a word list.

Although developed independently, if to be categorized the proposed analysis falls under the first category of approaches, related to Harris (1955).

However in contrast to the idea put forward by Harris, not the number of possible continuations of a substring is considered, but the predictability of the next character actually occurring in the text.

[1]Example text is written in *italic* and blanks are indicated by underscores.

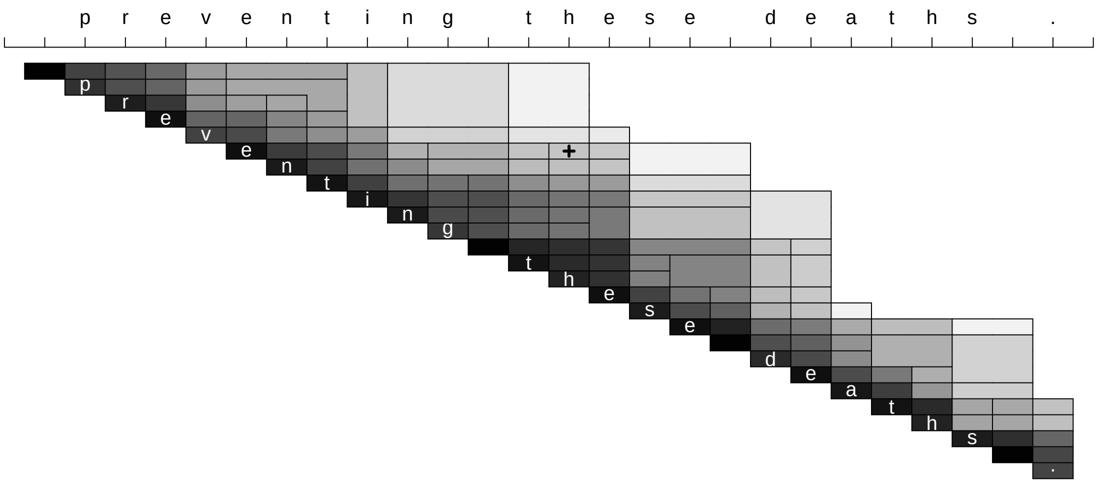

Figure 1: The raw frequency counts for the example chunk *_preventing_these_deaths_.*. Each colored field represents the frequency count for one particular substring: The field marked by the cross corresponds to the substring *enting_th*. Dark colors mean high frequencies. Frequency differences are indicated by solid lines between adjacent fields. These frequencies may well be so small that the grey scale is not fine enough to discern them.

These two quantities are inversely correlated, since a high successor variety implies a low predictability. But while successor variety does not depend on the next character actually occurring in a test corpus, the measure proposed here yields different results in different contexts. This flexibility is highly desirable.

Details of the algorithm are explained in the following section.

3 The algorithm

The underlying idea is very old and basic and reads as follows: “A unit is made up of parts which commonly occur together”.

The implementation of this idea can be stated as: “The text is segmented at points where the predictability of the next character drops”.

Starting point for the segmentation are the raw frequency counts for all substrings in a test corpus as they occur in a training corpus.

The frequency counts are easily accessible when stored within a suffix tree of the text. Fundamental knowledge of this data structure can for example be gained from Gusfield (1997).

Figure 1 pictures the raw frequency counts for an exemplifying chunk of data: *_preventing_these_deaths_.*[2].

At first glance these frequency data seem to contain meaningful patterns: With the naked eye one can discern the steps representing the substrings *_preventing_* and *_these_* . But the picture is not clear and a lot of filtering needs to be done to identify the linguistic units within the noise.

To find a segmentation of the text on the basis of the raw frequency counts, the concept of *predictability* is used. We start with a simple observation: The space after *_these* is very easy to predict: Figure 1 shows that 100% of the occurrences of *_these* were followed by a space. But after the space a drop occurs: only 18 out of the 385 occurrences of *_these_* or 4.7% were followed by the letter *d*. If we interpret these relative frequencies as an estimation of likelihood, we can use them to predict the next character. Since a drop in predictability occurs after *_these_*, a segment border is set there.

[2]There are spaces (underscores) before punctuation marks because I used a preprocessed version of the corpus for technical reasons. This does not affect the results, but it simply changes the definition of all punctuation to a sequence of space followed by the punctuation character.

The same argument can be used to find word or morpheme boundaries in the backward direction: 385 out of the 452 occurrences of *these_* (83%) in the training data are preceded by a space. But only 20 of these 385 or 5% are preceded by a *g*.

This concept leads to segmenting the text into substrings which have a drop of predictability in both directions at their borders. Let me state this more formally:

3.1 Definition of Predictability

Let $s_t(i,m) = t_i t_{i+1}..t_{i+m-1}$ be a character string of length m starting at character i within the test corpus $t = t_1 t_2..t_n$. Let T be the training corpus, and $N_T(s_t(i,m))$ the number of occurrences of $s_t(i,m)$ in T. Accordingly $N_T(t_{i-1}s_t(i,m))$ and $N_T(s_t(i,m)t_{i+m})$ are the training frequencies of the prolongations to the left and to the right of $s_t(i,m)$ by the characters preceding and succeeding it in the test corpus t.

I define

$$V^+_{T,t}(i,m) = \frac{N_T(s_t(i,m)t_{i+m})}{N_T(s_t(i,m))} \tag{1}$$

as the *forward-predictability* at character i and substring length m in the test corpus t relative to the training corpus T. The *backward-predictability* at i and m is accordingly defined as

$$V^-_{T,t}(i,m) = \frac{N_T(t_{i-1}s_t(i,m))}{N_T(s_t(i,m))} \tag{2}$$

The *forward predictability drop* at i and m is defined as:

$$D^+_{T,t}(i,m) = \frac{V^+_{T,t}(i,m)}{V^+_{T,t}(i,m-1)} \tag{3}$$

The *backward predictability drop* is then:

$$D^-_{T,t}(i,m) = \frac{V^-_{T,t}(i,m)}{V^-_{T,t}(i+1,m-1)} \tag{4}$$

All substrings $s_t(i,m)$ of t with $D^+_{T,t}(i,m) < 1$ and $D^-_{T,t}(i,m) < 1$ are considered possible segments.

Figure 2 shows the same data as figure 1 together with all possible segments as defined above.

3.2 Disambiguation

Figure 2 shows two very different things: First, the segmentation can be made hierarchical in the sense that segments can be split up into others: *_preventing_* can be divided in *_prevent* and *ing_*.

Secondly there are too many possible segments and they can be grouped into different hierarchical structures.

This makes disambiguation a necessity. Before different strategies are discussed in 3.2.2, the next section explains how a big fraction of wrong segment candidates can easily be ruled out from the start.

3.2.1 Completeness

Not only *_these_* constitutes a possible segment in figure 2, but also the spurious *_the*, obviously backed by the high frequency of the definite determiner in English texts.

But as can be seen, there is no possible segment starting at this position in the text, particularly, *se_* is no possible suffix in English morphology.

So we establish a constraint: Only those segments are considered, which can be arranged with other segments into a non-overlapping but complete partition of the text. This greatly reduces the set of candidate segments as can be seen in figure 2. From all possible segments, indicated by dots, only the white dots with black borders survive the constraint, the black dots with white borders miss it.

3.2.2 Disambiguation strategies

Disambiguation of the remaining segmentations can be split into two subtasks. I gave them **boldface** lower case names to distinguish them from the different strategies tested to solve them written UPPER CASE.

topdown: If we have a possible segment like *_preventing_* how shall we cut? Here solutions like (*_prevent*)(*ing_*) are to be preferred to the wrong (*_preventi*)(*ng_*) which is also sanctioned by figure 2.

follower: Where to go from a possible segment like *_these_*? Here we want to rule out the continuation with *_death* in favor for *_deaths_* which can be subdivided into (*_death*)(*s_*).[3]

A variety of different strategies was tested:

topdown. Each possible decomposition of a substring $z = s_t(i,j)$ into $(x)(y)$ with $x = s_t(i,m)$ and $y = s_t(i+m,n)$ with $(m+n=j)$ was scored by a

[3]Each strategy used to solve the **follower** task can be applied locally or globally. This is governed by two other parameters. We get back to this very soon.

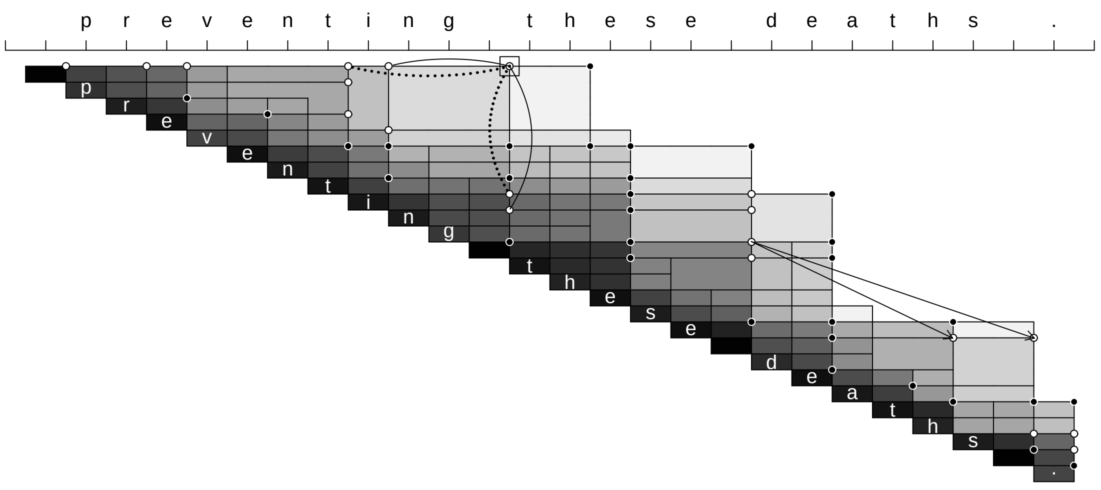

Figure 2: The same data as in figure 1. All possible segments are indicated by a dot in the upper right corner of the field. So the framed dot represents the possible segment *_preventing_*. The two ways of partitioning it into (*_prevent*)(*ing_*) and (*_preventi*)(*ng_*) are shown by solid and dashed arcs respectively. The two possible continuations after the segment *_these_* are referred to by the arrows, which end at *_death* and *_deaths_*. The segments which are ruled out from the outset by requiring a complete segmentation are shown as black dots with a white border, the remaining ones are inversely marked.

function S. The decomposition yielding a minimum S was chosen. The following definitions for S were tested:

PF Predictability in forward direction. The score for segmenting z into $(x)(y)$ is defined as $S_{PF} = -N_T(xy)/N_T(x)$, i.e. the fraction of occurrences of x in the training corpus which continue with y. The minus assures that a high predictability is rated as good.

PB Predictability in backward direction. $S_{PB} = -N_T(xy)/N_T(y)$, that is the fraction of occurrences of y that were preceded by x.

PDF Predictability drop in forward direction for the first segment x: $S_{PDF} = D^+_{T,t}(i,m)$

PDB Predictability drop in backward direction for the second segment y, that is $S_{PDB} = D^-_{T,t}(i+m,n)$

PD2 Logarithmic sum of the predictability drops in both directions for both segments or $S_{PD2} = \log(D^+_{T,t}(i,m)) + \log(D^+_{T,t}(i+m,n)) + \log(D^-_{T,t}(i,m)) + \log(D^-_{T,t}(i+m,n))$. The logarithm was chosen in order to make use of three of its properties: First, $\log(1) = 0$, that is no predictability drop gives zero score. Secondly $\log(0) = -\infty$, that is maximum predictability drop (which is unreachable), gets a maximum score. Third, when compared the same relative difference in D yields the same absolute distances in score: $\log(aD) = \log(a) + \log(D)$, independent of the value of D.

follower. For solving the task of selecting the next segment $y = s_t(i,m)$ to the right of a known suffix x, each possible y was priced with a score P. The following measures were considered for P

LONG Choose the longest y: $P_{LONG} = -m$ (minimizing favors the longest segment.)

SHORT Choose the shortest y: $P_{SHORT} = m$ (minimizing favors the shortest segment.)

FPDF Maximize the forward predictability drop: $P_{FPDF} = \log(D^+_{T,t}(i,m))$.

FPDB Maximize the backward predictability drop: $P_{FPDB} = \log(D^-_{T,t}(i,m))$.

FPD Combine both: $P_{FPD} = P_{FPDF} + P_{FPDB}$.

All these five strategies for solving the disambiguation subtask **follower** by defining some score function P can be applied locally or globally in two ways.

locglob: Only consider the immediately adjacent segment candidates or consider all following ones as well. Possible strategies are:

LOCAL Only use the P score of the directly following candidate segment y.

AV Compute the P score of y and of all of its followers until the end of the sentence and use the average of these values to judge y.

tdav: Finally there are different ways of computing the P score for a segment y which is itself split into two segments $(v)(w)$:

TOP Only compute the P score of y itself.

ALLAV Compute the P score of y, u, and v and of all of their parts recursively. Use the average as a final score for y.

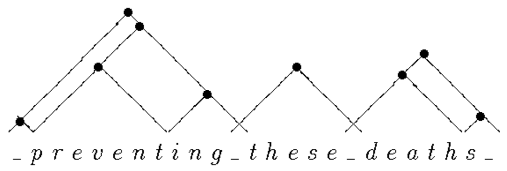

Figure 3: The final partitioning of the data. The example chunk is correctly segmented. The two spaces between *preventing* and *these* and between *these* and *deaths* belong to both segments each. The strategies used were PD2, FPDF, AV, and ALLAV.

Figure 3 shows the final decision made for the example chunk, if the procedures PD2, FPDF, AV, and ALLAV are used. As can be seen, the segmentation into $\{[(\_)($*prevent*$)]($*ing*$\_)\}(\_$*these*$\_)[(\_$*death*$)($*s*$\_)]$ coincides with the analysis of standard linguistics in this case. The whole phrase is not recognized as such. However, this is what we expect, since the algorithm at its present stage is only capable of detecting units which repeat.

3.2.3 Interim summary

To sum up so far, the algorithm divides the text into segments which have a drop in predictability at both of its borders. Afterwards, disambiguation procedures are run to find the best overall segmentation of the text.

Many of the segments can be discarded from the start. For the rest we need to score the segments and their relations. This task is broken down as far as possible and for the subtasks a variety of strategies are investigated.

Basically, only three minor linguistic prerequisites are hardwired into the system.

First: Natural language text can be split up into characters which follow each other in a row. While this cannot be said to be true for all scripts, it is definitely true for the two languages considered so far, English and German. For other cases, the method might have to be changed.

Secondly: Because the drop in predictability usually occurs before and after the spaces framing a graphical word, whitespace characters in contrast to visible characters can belong to two adjacent segments. The linguistically trivial distinction between visible and invisible characters is not known to the computer a priori.

Similarly, the third linguistic fact entering the model is the assumed identity of lower and upper case. If the system works in a case sensitive way, we get pseudo segments. To give an example, about 8% of the occurrences of the English definite determiner *the* are upper case in the used training corpus. For these we get a segmentation as (*T*)(*he*_). To avoid this, the program was run exclusively on lower case text.

4 Evaluation

4.1 The used measure

Evaluation in computational linguistics is usually done by giving *precision* p and *recall* r:

$$p = \frac{\text{number of correct decisions made}}{\text{number of decisions made}}$$
$$r = \frac{\text{number of correct decisions made}}{\text{number of possible correct decisions}}$$

These measures require the definition of a correct decision. This is not a problem as long as a correct decision can be defined as one which coincides with a given theory.

In the introduction we stated that the proposed algorithm aims at segmenting text into morphemes, words and phrases and at grouping these units according to their linguistic interrelations. For using recall and precision for evaluation we need the correct solutions to both of these problems. For this we need full-fledged theories of morphology and syntax. But even if we had such theories at hand, we would still evaluate in relation to theories, not to reality.

The reality we want to assess is the human language faculty. Of course we cannot do this directly, because we do not know in which units the human brain would break down the test corpus.

But there is a loophole left: It might be impossible to tell, whether the partition of *loophole* into (*loop*)(*hole*) is correct or if this string is better processed as a whole. But there should be no disagreement among native speakers of English that a segmentation into (*loo*)(*phole*) is definitely wrong. So even if we cannot honestly claim to be able to tell how many correct decisions the system makes, we are nevertheless able to tell how often it blunders. I tentatively define a *blunder* as an error no native speaker would ever make. While it would be desirable to have a stronger definition of this term, this preliminary one will suffice for the given study. We take the unclear cases (which will not be so many), as an estimation of the error margin.

This way of evaluation is not as much an embarrassing back door as it might seem. The perfect natural language processing system would not be the one making no errors at all, but the one which avoids errors a human would never make. Consequently, if we look at the bold blunders, we look at exactly the kind of error we have to worry about.

Resulting from these considerations the measure of *pseudoprecision* P_Ψ is defined for comparing the different disambiguation strategies:

$$P_\Psi = 1 - \frac{B}{B+O}\,, \qquad (5)$$

where B is the number of blunders and O is the number of non blunders.

The due objection that it is meaningless to work with *pseudoprecision* without defining *pseudorecall* must be taken seriously. In languages like English segmentation along white space will lead to very few blunders, even if it might be inappropriate to split up proper nouns like *New York*.

But for any definition of *pseudorecall* we would need to know the splits which all native speakers would set, but which the system did not set. To render such a definition precise enough to get reproducable results seems an intractable task.

Nevertheless, a natural language processing system which produces no blunders is either extremely good or trivial. We have to cope with the fact that recall always presupposes to know the full set of true answers which is not always a meaningful concept.

4.2 Evaluation procedure

Since there are five strategies for solving the **topdown** disambiguation subtask, five for **follower**s, and two for both **locglob** and **tdav**, there are altogether $5 \cdot 5 \cdot 2 \cdot 2 = 100$ different combinations. This is too much for manual evaluation.

So I resorted to a less complete procedure. I started with a plausible choice for three of the four parameters, and varied the fourth.

After this, each parameter is varied separately, starting from the optimal choice reached so far.

Two aspects of the segmentation of the test corpus were assessed separately: The points where cuts are set by the system (4.3) and the resulting segments delimited by these cuts (4.4). To give examples, (*loo*)(*phole*) is a blunder in the first sense, because the cut is clearly set at a wrong position, while (*prevent*)[(*ing_*)(*_these*)] is a blunder in the second sense, because the ending is attached to the next word wrongly.

The primary evaluation was done for German and for the quality of the cuts set by the system (4.3). As a result we get an optimal set of disambiguation strategies to solve the subtasks **topdown**, **follower**, **locglob**, and **tdav**. This optimal set is tested on English (4.3) and on the buildup of structure for both English and German (4.4).

For both training and testing different parts of the EUROPARL (Koehn, 2002) corpus were used. I used this parallel corpus to guarantee the comparability of the data for the two different languages. Training was done on a text of only about 1.300.000 characters. The data used for both languages were aligned translations of each other.

4.3 Cuts

If we fix **topdown**, **locglob**, and **tdav** to PD2, AV, and ALLAV respectively, we get the following performance values for the various strategies solving the **follower** disambiguation subtask:

follower	P_Ψ	Instances tested
LONG	0.862 ± 0.018	225
SHORT	0.748 ± 0.014	147
FPDF	0.861 ± 0.024	166
FPDB	0.894 ± 0.015	273
FPD	0.863 ± 0.015	172
	0.872 ± 0.017	450

The last table row shows that testing more instances yields consistent results.

Now **follower** was set to FPDB, both **locglob** and **tdav** were kept at AV and ALLAV, while the strategy for **topdown** was varied. The results are shown below.

topdown	P_Ψ	Instances tested
PF	0.874 ± 0.018	390
PB	0.891 ± 0.017	270
PDF	0.888 ± 0.016	273
PDB	0.881 ± 0.016	274
PD2	0.894 ± 0.015	273

Now, **topdown** was set to its optimum PD2 again and **locglob** and **tdav** were varied:

locglob, tdav	P_Ψ	Instances
LOCAL, TOP	0.802 ± 0.023	131
LOCAL,ALLAV	0.793 ± 0.022	135
AV, TOP	0.877 ± 0.021	409
AV, ALLAV	0.894 ± 0.015	273

The performance on the English test corpus with the winning set of strategies (**topdown**: PD2, **followers**: FPDB, **locglob**: AV, **tdav**: ALLAV) was 0.889 ± 0.015 with 239 instances tested, that is the results for German and English overlap greatly with respect to their error margins.

4.4 Linguistic units

In this second evaluation procedure it was checked if a segment with two non-blundering boundaries constitutes a possible linguistic unit which is a morpheme, word, or phrase. As above, no gold standard was set up, but the number of blunders like *detain*(*ed in the uk*) was counted to compute pseudoprecision as defined in equation 5.

The parameters were set to the optimal values as obtained in the preceding section: **topdown** to PD2, **locglob** to AV, and **tdav** to ALLAV, and **follower** to FPDB. Results were as follows:

Language	P_Ψ	Instances tested
German	0.893 ± 0.008	243
English	0.870 ± 0.008	311

Interestingly, most of the non-blundering segments are simply morpheme candidates. If only segments containing subsegments are counted, the figures fall to values nearer to 0.5 than to 1.

5 Discussion

There are two classes of results for the various disambiguation strategies tested in 4.3. Two very naïve strategies perform badly: Always continuing with the shortest segment (SHORT) is obviously suboptimal ($P_\Psi \approx 0.75$) and only scoring the very next segment (LOCAL) yields a P_Ψ around 0.8. All other strategies figure around a P_Ψ of 0.9 without significant differences. Nearly all of the remaining 10% blunders are avoided by at least one of the strategies (that is how they were set up), but the overall performance stays the same. This resembles very much the behavior found for state-of-the-art methods of unsupervised segmentation of words into morphemes which stop at an F measure between 0.6 and 0.7 (Kur, 2005). Similar figures are found in all papers cited in the present study.[4]

It should be mentioned that if only word internal cuts are considered[5] P_Ψ drops from 0.894 ± 0.015 to 0.812 ± 0.042 for the winning parameter setting for German (96 instances). For English the figure is dropping to 0.609 ± 0.064 (55 instances).

The demonstrated stability of the results should be worrying for everyone doing language structure learning purely on surface statistical basis. The remaining 10% blunders simply do not seem to stem from surface statistics.

Surely, some really promising possibilities to improve the results have not been applied yet: Parliament discussions are restricted in genre. One could try the method on a more balanced corpus. Moreover, the training corpus was very small. It only contains about 1.3m characters or around 300 pages of text, about a tenth of the small Brown corpus. Not even a cutoff was used to exclude segments with too bad statistics. It is – by the way – an astonishing fact, how good the results are from this point of view and how often segments were recognized from one occurrence in the training corpus only (*lord_inglewood* is an example).

The most interesting and so far untested way of excluding spurious segments by the use of context information is sketched in figure 4.

But apart from these prospects a glance at the occurring blunders shows two things: First, many wrong segmentations clearly stem from the complete lack of categorical knowledge in the system: In German the occurring segmentation [(*ver*)(*sich*)]{[(*er*)(*ung*)](*en*)}[6] is caused by the reflexive and personal pronouns *sich* and *er* which

[4] Of course no quantitative comparison is possible between my and their results, because I refrain from using precision and recall, which are usually given.

[5] That is if all cuts which coincide with white space are excluded.

[6] instead of {[(<*ver*><*sicher*>)(*ung*)](*en*)}, assurances

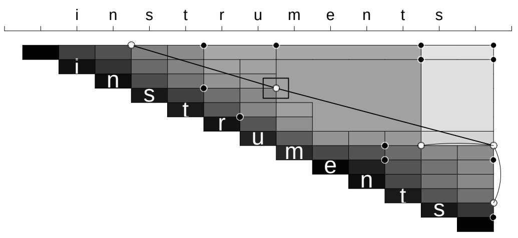

Figure 4: The segment represented by the framed dot (*stru*) is definitely no possible morpheme in this context. It is a blunder of the current setup. But as shown in the figure, every occurrence of *strument* was preceded by an *in*. This fact should help to suppress the wrong segment.

could be ruled out in this context if recognized as such.

Secondly, many wrong segment groupings such as the German (*wesentlich*)[(*er*)(_*teil*)][7] where the inflectional ending is attached to the following noun instead of the adjective have their origin in syntactic agreement: Because the adjective ending is determined by the noun, its affiliation to the adjective is not resolvable statistically. Here we also need categorical information.

This is why I strongly suggest to give up the long standing paradigm of sequential processing, where preprocessing is followed by tokenization is followed by tagging is followed by grammatical analyzes. It should be replaced by a more holistic approach which is capable of recognizing linguistic categories while processing raw text. This could consist of a combination of a system like the investigated and an approach as it is described in Solan et al. (2005).

6 Acknowledgments

I wholeheartedly thank my doctoral adviser Anke Lüdeling for her outright support, and her and Bettina Schrader, Stefan Evert, Ulf Leser, and Karsten Tabelow for enlightening discussions and Mascha, Verena, and Frank for invaluable proofreading.

References

Stefan Bordag. 2005. Unsupervised, knowledge-free morpheme boundary detection. In *Proceedings of RANLP 05*, Borovets.

M. Creutz and K. Lagus. 2005. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. Publications in Computer and Information Science Report A81, Helsinki University of Technology, March.

Minh Thang Dang and Saad Choudri. 2005. Simple unsupervised morphology analysis algorithm (sumaa). In *Unsupervised segmentation of words into morphemes - Challenge 2005*. CIS – Laboratory of Computer and Information Science.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Comput. Linguist.*, 27(2):153–198.

Dan Gusfield. 1997. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press.

Margaret A. Hafer and Stephen F. Weiss. 1974. word segmentation by letter successor varieties. *Inform. Stor. Retr.*, 10:371–385.

Harald Hammarström. 2006. A naive theory of morphology and an algorithm for extraction. In *SIGPHON-06*.

Zellig S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222. reprinted in (Hiż, 1970).

H. Hiż, editor. 1970. *Papers in Structural and Transformational Linguistics*. Dordrecht, Holland.

P. Koehn. 2002. Europarl: A multilingual corpus for evaluation of machine translation. `http://people.csail.mit.edu/~koehn/publications/europarl.ps`. Draft, unpublished.

Mikko Kurimo, Mathias Creutz, and Krista Lagus, editors. Unsupervised segmentation of words into morphemes – challenge 2005 [online, cited Thu Apr 27 2006]. Available from: `http://www.cis.hut.fi/morphochallenge2005/fmeasure.shtml`.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. *Proc Natl Acad Sci U S A*, 102(33):11629–34, August.

[7] *essential part*