Studies for Segmentation of Historical Texts: Sentences or Chunks?

Florian Petran

Department of Linguistics
Ruhr-Universität Bochum
E-mail: petran@linguistics.rub.de

Abstract

We present some experiments on text segmentation for German texts aimed at developing a method of segmenting historical texts. Since such texts have no (consistent) punctuation, we use a machine learning approach to label tokens with their relative positions in text segments using Conditional Random Fields. We compare the performance of this approach on the task of segmenting of text into sentences, clauses, and chunks, and find that the task gets easier, the smaller grained the target segments are.

1 Introduction[1]

Text segmentation is an important part of the natural language processing pipeline. Not only is it an important target for corpus annotation, since the sentence and constructions on the sentence level are subject to research, it is also an important pre-processing step for a lot of other annotations that presuppose a segmented text. Tasks such as bitext alignment or part-of-speech tagging either require a segmented text, or work better or faster with pre-segmentation.

Segmenting text into sentences is usually seen as a task of disambiguating punctuation marks, and there are several systems that perform that task well. But what if a text has no or no consistent punctuation to indicate sentence boundary? Syntactically motivated punctuation was not used until well in Early Modern times for European languages, or even as late as the 19th century for Chinese texts. Anyone working with historical texts will therefore be faced with the problem of having to manually annotate for sentence segmentation which is a non-trivial, time consuming task. Spoken language faces similar problems as well, but it has intonation features as indicators of sentence breaks, something which is largely absent in historical texts.

[1]The research reported here was financed by Deutsche Forschungsgesellschaft (DFG), Grant DI 1558/4-1.

There is already some related work on Chinese sentence segmentation, and spoken language segmentation (see section 2 below). Based on those results, this paper introduces experiments with segmentation of modern German language data using Conditional Random Fields (Lafferty et al. [5]) without the use of punctuation marks. They are aimed at developing a method of segmenting historical texts. In the results, we find that a chunking approach works far better than the sentence segmentation, which suggests that it would be prudent to do the sentence segmentation based on the chunks determined.

The paper is organized as follows. Section 2 introduces approaches this work is based on. The data used for the experiments, and the method used to obtain it is described in section 3. Section 4 describes the experiments, their results, and the method of evaluation. Finally, section 5 offers some concluding thoughts and directions for future research.

2 Related Work

Most sentence segmentation systems for Western languages rely on the presence of periods in the text. These need to be disambiguated between sentence terminating punctuation symbols, and punctuation in other functions such as abbreviations. In that, systems such as Gillick [2] achieve very good results with F-scores well over 90%. However, the solutions are not transferrable to a situation where punctuation is inconsistent, or entirely absent.

One such situation is spoken language data, where capitalization or punctuation do not exist at all. For such cases, a machine learning approach is used to classify tokens according to whether they constitute a sentence boundary or not (such as in Stevenson and Gaizauskas [10]). More recent approaches achieve good results with Hidden Markov Models or Conditional Random Fields (Liu et al. [6],[7]), but they use prosodic information such as timing, energy, or pitch as additional features. With the NIST SU detection data set, there is an established gold standard data set that is used by many researchers. No such data set exists for historical or modern German.

One of the key features of speech data is that it is not always straightforward to segment into sentences, since a sentence in spoken language is not as well defined as in written language: due to their spontaneous nature, spoken sentences may be interrupted, or syntactically incomplete. Furthermore, even in written language it is often subjective if a comma or a period should be used at a given point in the text. For research on spoken language data, this is approached with the definition of sentential units that are annotated in the transcribed data instead. An inter-annotator agreement of 93% for the NIST SU detection task (Liu et al. [6]) shows that even the manual annotation is not a trivial task. Spoken language data bears some similarity to historical texts in that the transcription is another potential source of error, whereas historical language has the normalization of spelling variants as error source. However, the key difference is that prosodic information, used by all newer approaches to speech segmentation, is not available for historical texts.

Another important difference is that spoken language research focuses mostly on English, which has a relatively fixed word order. The results are thus not entirely transferrable to a language with more free word order, such as German. Nevertheless, an important result from Liu et al. [7] that influenced the experiments presented here is the improved performance of a CRF based system compared to HMM or MaxEnt based ones.

In Chinese, punctuation was not commonly used until well in the 19th century, and is often not used today in informal (Internet) communication (Huang et al. [3]). Huang et al. [4] employ a setup very similar to the one used in this paper, where CRF are used to label different data sources, and test different tagsets for sentence breaks. In addition to POS tags they use phonetic representation of the words indicating (among other things) the initial and final tone. It is worth noting that even though they achieve F-scores of up to 83% on historical texts, modern Chinese texts tend to have more complex sentences, and are therefore harder to segment and they achieve far lower results on them. Obviously, the segmentation task is easier, the simpler the units are.

3 Data Basis and Extraction Methods

So far, there are no complete historical treebanks of German that have suitable annotation for all tasks. Instead, the data used for the experiments is extracted from the Tübinger Baumbank des Deutschen/Zeitungstext (TübaDZ)[2], a treebank of about 65,000 sentences of German newspaper texts. Modern German uses capitalization to indicate nouns, but this is usually not the case for Early Modern German. Additionally, an important property related to the task is the absence of (consistent) punctuation. We therefore filtered the punctuation marks from the texts and downcased all tokens, so as to simulate a historical text as closely as possible. Another important property of historical texts is frequent occurence of variant spelling, which was not simulated. Instead, we assume a work environment where all historical tokens are already mapped to their modern equivalent spellings.

3.1 Data Source and Sentence Segmentation Task

For the annotation of boundaries, we used the 5-tags set that performed best for the Chinese segmentation task (Huang et al. [4]), described in Table 1.

As far as data extraction is concerned, the simplest possibility is to annotate the boundaries of a sentence (S node) in the data. For this annotation we need not take the parse tree into account and use no additional heuristics.

[2]http://www.sfs.uni-tuebingen.de/tuebadz.shtml

Tag	Meaning
L1	Beginning of segment
L2	Second token in segment
M	Middle of segment
R	End of segment
S	Singleton segment

Table 1: 5-tags set for sentence segmentation.

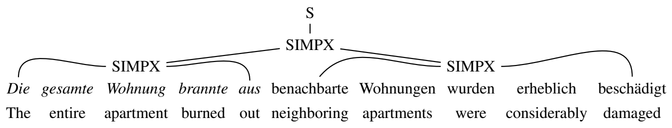
S
SIMPX
SIMPX
SIMPX
Die gesamte Wohnung brannte aus benachbarte Wohnungen wurden erheblich beschädigt
The entire apartment burned out neighboring apartments were considerably damaged

Figure 1: "The entire apartment burned out, neighboring apartments were severely damaged." — An example for embedded SIMPX with a straightforward solution.

3.2 Clause Extraction Task

Due to the recursive nature of clauses, their extraction from TübaDZ is not as straightforward. The TübaDZ parse tree is annotated with categories for simple clauses, relative clauses, and paratactic constructions of simple clauses, all of which are collectively referred to as SIMPX from here on. The extraction of these requires some heuristics, which are further described on some simplified example parse trees. First, a SIMPX node can itself contain other SIMPX nodes as the simplified parse tree in Fig. 1 shows. For these cases, we decide to always use the lowest SIMPX nodes, i.e. those further away from the root of the tree, and discard the higher level clause for segment annotation purposes.

Since German has a relatively free word order, this heuristics is not in all cases sufficient, as the example of a partial sentence in Fig. 2 shows. The simple clause begins at the first token *verantwortlich*, and then continues from *sei* until the end of the example. Beginning at the second token *so* up to *AWO*, however, is a paren-

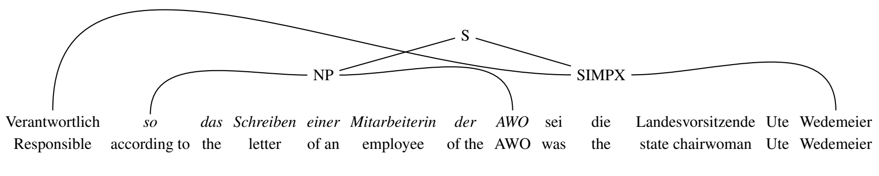
S
NP
SIMPX
Verantwortlich so das Schreiben einer Mitarbeiterin der AWO sei die Landesvorsitzende Ute Wedemeier
Responsible according to the letter of an employee of the AWO was the state chairwoman Ute Wedemeier

Figure 2: "According to the letter of an employee of the AWO, state chairwoman Ute Wedemeier was responsible."—An example of an embedded structure that is not a simple clause by itself.

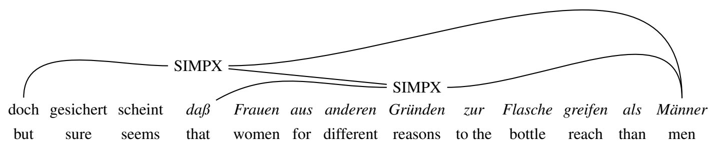
SIMPX
SIMPX
doch gesichert scheint daß Frauen aus anderen Gründen zur Flasche greifen als Männer
but sure seems that women for different reasons to the bottle reach than men

Figure 3: “but it seems safe to say that women take to the bottle for different reasons than men.” — A SIMPX node that contains another SIMPX, but also terminals.

thetical NP that is independent of the simple clause, and therefore directly attached to the root sentence node S. In this case, the heuristics of taking the smallest node obviously does not suffice, since both nodes are siblings. So we have elements of the sentence that are not part of any clause from a syntactic point of view.

For these cases we would extract the NP in the example as a clause on its own. The way this was implemented was to gather all terminal symbols that were not extracted as part of a clause, and regard them as a segment as long as they were consecutive. Furthermore, the flat boundary annotation scheme is unable to account for non-consecutive tokens in a segmental unit. Since we cannot reorder the data, we extract the SIMPX in cases like these as two different segments, one starting at *sei* up to the end, and one with *verantwortlich* as its only word.

Fig. 3 shows another (partial) example sentence from the corpus. If we apply the rule to always use the smallest SIMPX in this case, we would be left with unassigned terminal symbols in the result, since the SIMPX starting at *daß* running up to the end is embedded into the superordinate SIMPX, but there are also terminal symbols. In order to not overcomplicate the heuristics, we applied the rules set above straightforwardly in this case as well. This means that we extract all terminals preceding the smaller SIMPX as a clause on their own, and the following SIMPX as an independent clause. Overall, this means that the clause annotations are less than perfect, since the nature of a parsed corpus makes it ill suited for a shallow segmentation task.

3.3 Chunk Segmentation Task

Finally, we extracted chunks to compare the chunk segmentation task with the sentence segmentation. Unfortunately, again, the TübaDZ annotation is not always well suited for a chunking task. Although there are node labels for verb phrases, these are in fact used to annotate the verbal head of the phrase instead of the phrase governed by the verb. We therefore excluded verb phrases from the extraction and added another tag O to the tagset, indicating that a token is outside of an annotated chunk. The chunks were extracted in a similar way to the SIMPX nodes, preferring bottom most nodes for the extraction, with the following additions to the heuristics.

If a prepositional phrase was present, it was preferred over a noun phrase, and if a noun phrase was present, it was preferred over an adjective phrase. The only other phrases that are annotated in TübaDZ are complex determiners and certain adverbial phrases, but since these are always embedded in a predictable way, we ignored them for the data extraction.

3.4 Data Properties

Table 2 shows the tag distributions for each task. Obviously, the number of L1 and R tags is always the same because a segment starting with an L1 tag has to be always closed by an R tag. The difference between the amount of tokens tagged L1 and L2 indicates the amount of segments that are longer than two tokens: a two token segment only has L1 and R tags, and only segments with three or more tokens have the L2 tag. The difference between the amount of L2 and M tagged tokens indicate the amount of segments that are longer than three tokens, because only those would have the M tag, but also how likely it is that these segments are exactly four tokens, or longer.

	L1	L2	M	R	S	O
Sentence	6.4954%	6.2716%	**80.5324%**	6.4954%	0.2052%	–
Clause	14.1560%	12.5402%	**57.3251%**	14.1560%	1.8226%	–
Chunk	19.1940%	11.1127%	18.8499%	19.1940%	9.1268%	**22.5482%**

Table 2: Relative frequencies of the boundary tags in % of 832,975 tokens. Most frequent tags for each task are bold face. The leftmost column indicates the task.

For the sentence segmentation task, the number of L1 and L2 tags is roughly the same, which indicates that there are only few two token segments—incidentally about the same number as one token segments (tagged S). The vast majority of tokens is tagged M, which indicates that most segments are well over four tokens long. For the clause segmentation task, the number of two token segments is, again, roughly the same as the number of one token segments, but there are a lot more of both categories. Compared to the sentence task, there is a lower proportion of M tags, again unsurprisingly, as the segments are relatively shorter, since they are subsegments of sentences. The chunk segmentation task has a much more even distribution of tags, with a small majority of tokens tagged O (for outside of chunk). The relative numbers of L1 and L2 tags indicate that a little less than half the segments are only two tokens long. But for those that are at least three tokens long, there is a high probability that they are either four tokens or longer.

4 Results and Evaluation

4.1 Evaluation Method

All the following results were obtained using 5-fold cross validation, averaging the results over all folds. The corpus was split along the sentence boundaries closest to one fifth of the tokens, such as to minimize noise in the data. Training and application of the model was done using the CRF++ toolkit[3] with the default feature template, token and POS tag as feature, a minimum feature frequency of 2 and the default cost function of 1.0. All test and training data was converted to lower case.

In addition to testing with the gold standard POS data, we also annotated the test data with the TreeTagger (Schmid [9]) which ships with a parameter file for German. This was done to more closely simulate real world conditions, where text segmentation is hardly done on texts with manually corrected POS tags. Since the parameter file for the tagger expects mixed case input, we used the original, mixed case tokens for tagging and downcased them afterwards. However, explorative experiments suggest that tagging performance loss by retraining a German tagger on downcased data is negligible.

With a set of less than six tags, tagging accuracy has a fairly high baseline for the most frequent tag assumption. Evaluating tag accuracy as a percentage of correct tags is therefore not a very meaningful indicator of the overall performance of the system. Instead, we evaluated precision and recall for the segment boundaries in the text. This was determined as follows. If a token was tagged as the start of a segment, and if the immediately preceding token was tagged as the end of a segment, a boundary was assumed to be present between the tokens. Note that this means that the evaluation for the sentence tagset is theoretically slightly stricter than for the IOB tagset, since it relies on two tags being correct rather than just one. In practice, this does not matter so much, since the system reliably learned the correct sequence of boundary tags in all cases. A baseline for this evaluation approach is difficult to determine, and any random guessing baseline would be fairly low. For the purpose of this paper, however, it does not matter as much, since our focus is on the comparison of relative difficulties of the different tasks.

Where applicable, O–tagged tokens were counted in the same way as singleton phrases. Of course this means that a theoretical baseline is different for all tasks below (see section 3.4). The smaller the units that are tagged, the less M tagged tokens and the more S and O tagged ones appear. But none of the correct M tags count towards the performance reported below, since they are always negative categorizations.

4.2 Sentence Segmentation Task

The results for the sentence segmentation task are shown in Table 3. They are better than could be expected by random guessing, but certainly not good enough for real world application in this form. It seems that this task is especially sensitive to only slightly erroneous POS tags, so we performed an additional experiment with a simpler template using only the surface token as feature. If we compare those results to the results with generated tags, we notice, again, a marked decrease in overall performance, but not as much as from the gold POS tags to the generated

[3]http://crfpp.googlecode.com/

tags. In fact, the precision is even slightly higher without the tags.

	Precision	Recall	F-score
Gold POS Tags	71.57%	59.13%	64.76%
Generated POS	63.27%	48.72%	55.04%
Without POS	63.75%	40.15%	49.27%

Table 3: Results for the sentence extraction task.

The likely reason for the poor performance of this task is that sentence boundaries are, to some degree, subjective. It is often a matter of style if a writer uses a full stop, a comma, or a semicolon at a given point. This has a detrimental effect mainly on recall, as it is difficult to find all cases that had full stops in the original text, but it also effects the overall performance negatively, because many items in the training data are learned as in the middle of a sentence when they might as well have been the first token in a sentence. The comparatively higher precision suggests that a bootstrapping approach could possibly improve performance. The small effect of the generated POS tagging indicates that this might work even for a text where tagging proves difficult and could, in fact, be potentially used to improve tagging performance. But before such enhancements are considered, a more detailed error analysis would have to be done.

4.3 Clause Segmentation Task

For the simple clause boundary annotation task shown in Table 4, the performance is much better. This is especially remarkable given the tradeoffs from our clause extraction heuristics described in Sec. 3. Obviously, simple clause boundaries are easier to learn than sentence boundaries, even with noisy data. The noise also seems to affect recall slightly more than precision. In a real world application, such a task might benefit from annotation of sentential units instead of clauses or sentences, such as described in LDC [1].

	Precision	Recall	F-score
Gold POS Tags	83.00%	69.37%	75.57%
Generated POS	73.60%	58.67%	65.29%
Without POS	71.07%	54.44%	61.65%
Simplified data	88.25%	79.46%	83.72%

Table 4: Results for the clause extraction task.

The last line of Table 4 shows the results with gold tags for an idealized setting were only continuous SIMPX consisting of nothing but terminals were extracted. Note that this is not entirely comparable to the other rows, since the source data was adjusted, but it shows how performance might improve for an ideal language that did not force difficult decisions such as the ones described in Sec. 3 upon us. The effect this has on recall is roughly twice as high as the effect on precision. This shows that even for the noisy data, the system learned the right clues reliably, but was less able to apply them in all cases.

4.4 Chunk Segmentation Task

For the chunking task, we did an additional run of the experiments with the IOB tagset commonly used for chunking (Ramshaw and Marcus [8]). The IOB tagset only knows annotations for a position in- or outside of a chunk (I and O respectively), or starting a new chunk (B). The experiments done by Huang et al. [4] suggest that the 5-tags set performs best, but they do not evaluate its performance on chunking, or the IOB tagset.

As Table 5 shows, the chunking task yields very good results for both tagsets. The 5-tags set task outperforms the IOB tagset by a small margin. As the quality of the POS annotation decreases, however, the gap between the two annotation schemes shrinks slightly. It is also noteworthy, that the chunking task is far less sensitive to the quality, or in fact the presence of POS annotation than the other tasks, for both annotation schemes, and even slightly less so for the sentence tagset.

	5 tagset			IOB tagset		
	Precision	Recall	F-score	Precision	Recall	F-score
Gold POS Tags	92.72%	94.90%	93.79%	91.82%	94.77%	93.27%
Generated POS	91.19%	90.78%	90.98%	90.37%	92.70%	91.52%
Without POS	90.79%	88.87%	89.82%	88.36%	89.07%	88.71%

Table 5: Results for the chunking task using the extended 5-tags set and the IOB tagset, and gold or generated POS data respectively.

Of the three tasks examined here, the chunking task is the only one where recall is higher than precision, if only by a small margin, but recall is also more affected by the POS annotation quality than precision. In any case, and for both annotation schemes, the chunking task yields good results even with non-optimized, standard settings for the learning program.

4.5 Error Analysis

Tables 6, 7, and 8 show the confusion matrices for the sentence, clause, and chunk segmentation tasks respectively. All matrices were generated for the results with gold POS tags. They show that all tasks perform far better a random guessing scenario, and for almost all tags and tasks, the correct guess is the vast majority. For the sentence and chunk segmentation tasks, all tags have a relatively high probability of confusion with the M tag, which is to be expected since it is by far the most frequent in those tasks. The low probability of confusion of the L1, L2, and R tags with each other show that where the system guessed the boundary wrong, it was not off by just one or two positions, but generally did not recognize that a boundary was present at all, tagging M instead. So far, this is exactly what should be expected.

		Expected				
		L1	L2	M	R	S
Actual	L1	**31514 (58.78%)**	535 (1.04%)	11972 (1.01%)	349 (0.65%)	378 (22.13%)
	L2	646 (1.20%)	**30114 (58.60%)**	11816 (1.79%)	867 (1.63%)	44 (2.58%)
	M	20872 (38.93%)	20058 (39.03%)	**624645 (94.62%)**	20420 (38.29%)	497 (29.10%)
	R	389 (0.73%)	674 (1.31%)	11650 (1.76%)	**31630 (59.32%)**	204 (11.94%)
	S	195 (0.36%)	9 (0.02%)	55 (0.01%)	58 (0.11%)	**585 (34.25%)**

Table 6: Confusion matrix for the sentence segmentation task. Cells indicate the number of guesses for the category denoted by the row, the percentage is of the row total. Correct results are bold faced.

		Expected				
		L1	L2	M	R	S
Actual	L1	**82300 (70.36%)**	2772 (2.69%)	12283 (2.62%)	1908 (1.64%)	2991 (20.05%)
	L2	4052 (3.46%)	**73995 (71.82%)**	118818 (2.52%)	3637 (3.12%)	461 (3.09%)
	M	27232 (23.28%)	23907 (23.21%)	**431551 (92.07%)**	27447 (23.56%)	3919 (26.27%)
	R	1768 (1.51%)	2217 (2.15%)	12256 (2.61%)	**82614 (70.90%)**	3036 (20.35%)
	S	1626 (1.39%)	134 (0.13%)	825 (0.18%)	915 (0.79%)	**4510 (30.23%)**

Table 7: Confusion matrix for the clause segmentation task.

Concerning the singleton segment S tag for the sentence task, if we disregard the confusion with M, the system fairly frequently confused it with L1. This indicates that it often correctly identified a sentence starting at that position, but incorrectly guessed the length of that sentence. Overall, the S tag has the lowest accuracy in that task, probably because it is very rare in the training corpus. The difference is even smaller for the clause segmentation task, where the correct guesses for S only barely outweigh the L1 guesses. This is in spite of the fact that the S tag for clauses is far more frequent than for sentences, though still comparatively rare. The likely source of that problem are the problems with data extraction for clauses described in Sec. 3.2. Since a lot of S tagged tokens are not actually singleton segments, but instead part of discontinuous clauses, it was difficult to learn them reliably. This is confirmed by the far better results for this task on the idealized data as described in the bottom row of Table 4.

		Expected					
		L1	L2	M	R	S	O
Actual	L1	**112679 (88.92%)**	1775 (2.48%)	10166 (8.32%)	227 (0.18%)	1190 (2.00%)	1233 (0.82%)
	L2	1359 (1.07%)	**60486 (84.35%)**	6676 (5.46%)	2576 (2.05%)	161 (0.27%)	328 (0.22%)
	M	8544 (6.74%)	5579 (7.78%)	**88104 (72.11%)**	7966 (6.32%)	1401 (2.35%)	2143 (1.43%)
	R	200 (0.16%)	2906 (4.05%)	10506 (8.60%	**109891 (87.25%)**	2145 (3.60%)	944 (0.63%)
	S	1343 (1.06%)	223 (0.31%)	2281 (1.87%)	2161 (1.72%)	**53946 (90.66%)**	67 (0.04%)
	O	2600 (2.05%)	742 (1.03%)	4440 (3.63%)	3131 (2.49%)	663 (1.11%)	**145220 (96.86%)**

Table 8: Confusion matrix for the chunk segmentation task.

The confusion matrix for the chunk segmentation task shows the lowest accuracy for the M tag, which is mildly surprising since it is not all that infrequent in the data. On the other hand, the tag frequencies are much more balanced overall for this task, so any effect of tag frequency would be smaller overall. One factor that is not accounted for at all by the results reported in Sec. 4.4 is the remarkably small confusion between O and S. Any such confusion would not be counted as word boundary error by our evaluation scheme described in Sec. 4.1, since they both serve the same function with regards to the word boundary. This indicates that a possible labeling of chunks with or after the segmentation could work well.

5 Conclusion

We have shown that it is generally possible to segment German text without the use of punctuation marks. As the target units for segmentation become smaller, this task becomes easier, to a point where it achieves results that would be suitable as basis for manual correction in a real world application. Further improvements might be achieved by optimizing the parameters for the CRF system. The reason for this performance increase is likely that sentences are, to a degree, subjective, and their segmentation dependent on individual style in many cases. Clause units have fairly complex relations with each other and are therefore not easily captured by a flat boundary annotation scheme.

Chunks on the other hand, as defined by the data extraction in Sec. 3, are simple constituents of sentences that are easily objectively identifiable. This suggests that instead of annotating sentences, it would be more efficient to first chunk the texts, and then use those chunk boundaries to identify sentential unit breaks. Since the chunks can never cross sentence boundaries, the sentence segmentation based on the chunk boundaries would be reduced to the disambiguation of chunk boundaries, similar to the way that written text is normally segmented. This is a far easier task than segmentation of text without any indications of boundaries since it limits the possibilities of where boundaries can be located.

One direction that needs to be explored for future research is the performance of the system on an actual historical text. As mentioned above, these types of text have large amounts of variant spellings that need to be normalized for a model trained on modern German to be applicable. Such a normalization step would introduce an additional source of error to both POS tagging and the segmentation, so it needs to be clarified how large the impact of an additional error source on the overall system will be. Alternatively, the segmentation model could be trained on non-normalized historical data. For this case, the effect on training corpus size would need to be explored.

As an intermediate step, it would also be helpful to determine the performance on a corpus that has annotations for verb chunks as well. We had to leave these out of the experiments entirely because the TübaDZ annotations do not allow an extraction of verb chunks. Had VP annotation been present, we would have less out of chunk elements, less word boundaries, and likely longer segments overall.

Furthermore, there are various improvements that could be made on the performance of the system itself. For example, Liu et al. [7] report that a majority vote between MaxEnt, HMM, and CRF classifiers performed better than each individual one for the task of spoken language segmentation. Such a majority vote system would be computationally expensive, but not very hard to implement. Another possibility is a bootstrapping approach, suggested by the fact that for most tasks, precision is far higher than recall.

Finally, another open question is the labeling of chunks, which would, ultimately, be a desirable outcome of the chunk segmentation. There are basically two different approaches to the shallow parsing task with regards to our segmentation system. We could label the chunks after they have been segmented, or combine the chunk labeling with the segmentation for a larger tagset. Each approach could have its advantages, but discussing them is beyond the scope of this paper.

References

[1] Linguistic Data Consortium. Simple metadata annotation specification version 6.2. 2004.

[2] Dan Gillick. Sentence boundary detection and the problem with the US. In *Proceedings of NAACL*, pages 241–244. Association for Computational Linguistics, 2009.

[3] Hen-Hsen Huang and Hsin-Hsi Chen. Pause and stop labeling for chinese sentence boundary detection. In *Proceedings of RANLP*, pages 146–153, 2011.

[4] Hen-Hsen Huang, Chuen-Tsai Sun, and Hsin-Hsi Chen. Classical chinese sentence segmentation. In *Proceedings of CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 15–22, 2010.

[5] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

[6] Yang Liu, Nitesh V. Chawla, Mary P. Harper, Elizabeth Shriberg, and Andreas Stolcke. A study in machine learning from imbalanced data for sentence boundary detection in speech. *Computer Speech & Language*, 20(4):468–494, 2006.

[7] Yang Liu, Andreas Stolcke, Elizabeth Shriberg, and Mary Harper. Using conditional random fields for sentence boundary detection in speech. In *Proceedings of the 43rd Annual Meeting of ACL*, pages 451–458. Association for Computational Linguistics, 2005.

[8] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94. ACL, 1995.

[9] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, volume 12, pages 44–49, 1994.

[10] Mark Stevenson and Robert Gaizauskas. Experiments on sentence boundary detection. In *Proceedings of the sixth conference on Applied natural language processing*, pages 84–89. ACL, 2000.