Tagging Sentence Boundaries

Andrei Mikheev
Xanalys Inc. and The University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, UK
mikheev@harlequin.co.uk

Abstract

In this paper we tackle sentence boundary disambiguation through a part-of-speech (POS) tagging framework. We describe necessary changes in text tokenization and the implementation of a POS tagger and provide results of an evaluation of this system on two corpora. We also describe an extension of the traditional POS tagging by combining it with the document-centered approach to proper name identification and abbreviation handling. This made the resulting system robust to domain and topic shifts.

1 Introduction

Sentence boundary disambiguation (SBD) is an important aspect in developing virtually any practical text processing application – syntactic parsing, Information Extraction, Machine Translation, Text Alignment, Document Summarization, etc. Segmenting text into sentences in most cases is a simple matter – a period, an exclamation mark or a question mark usually signal a sentence boundary. However, there are cases when a period denotes a decimal point or is a part of an abbreviation and thus it does not signal a sentence break. Furthermore, an abbreviation itself can be the last token in a sentence, in which case its period acts at the same time as part of this abbreviation and as the end-of-sentence indicator (fullstop).

The first large class of sentence boundary disambiguators uses manually built rules which are usually encoded in terms of regular expression grammars supplemented with lists of abbreviations, common words, proper names, etc. For instance, the Alembic workbench (Aberdeen et al., 1995) contains a sentence splitting module which employs over 100 regular-expression rules written in Flex. To put together a few rules which do a job is fast and easy, but to develop a good rule-based system is quite a labour consuming enterprise. Another potential shortcoming is that such systems are usually closely tailored to a particular corpus and are not easily portable across domains.

Automatically trainable software is generally seen as a way of producing systems quickly re-trainable for a new corpus, domain or even for another language. Thus, the second class of SBD systems employs machine learning techniques such as decision tree classifiers (Riley, 1989), maximum entropy modeling (MAXTERMINATOR) (Reynar and Ratnaparkhi, 1997), neural networks (SATZ) (Palmer and Hearst, 1997), etc.. Machine learning systems treat the SBD task as a classification problem, using features such as word spelling, capitalization, suffix, word class, etc., found in the local context of potential sentence breaking punctuation. There is, however, one catch - all machine learning approaches to the SBD task known to us require labeled examples for training. This implies an investment in the annotation phase.

There are two corpora normally used for evaluation and development in a number of text processing tasks and in the SBD task in particular: the Brown Corpus and the Wall Street Journal (WSJ) corpus – both part of the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993). Words in both these corpora are annotated with part-of-speech (POS) information and the text is split into documents, paragraphs and sentences. This gives all necessary information for the development of an SBD system and its evaluation. State-of-the-art machine-learning and rule-based SBD systems achieve the error rate of about 0.8-1.5% measured on the Brown Corpus and the WSJ. The best performance on the WSJ was achieved by a combination of the SATZ system with the Alembic system – 0.5% error rate. The best performance on the Brown Corpus, 0.2% error rate, was reported by (Riley, 1989), who trained a decision tree classifier on a 25 million word corpus.

1.1 Word-based vs. Syntactic Methods

The first source of ambiguity in end-of-sentence marking is introduced by abbreviations: if we know that the word which precedes a period is *not* an abbreviation, then almost certainly this period denotes a sentence break. However, if this word is an abbreviation, then it is not that easy to make a clear decision. The second major source of information for approaching the SBD task comes from the word which follows the period or other sentence splitting punctuation. In general, when the following word is punctuation, number or a lowercased word – the abbreviation is not sentence terminal. When the following word is capitalized the situation is less clear. If this word is a capitalized common word – this signals start of another sentence, but if this word is a proper name and the previous word is an abbreviation, then the situation is truly ambiguous.

Most of the existing SBD systems are word-based. They employ only lexical information (word capitalization, spelling, suffix, etc.) to predict whether a capitalized word-token which follows a period is a proper name or is a common word. Usually this is implemented by applying the lexical lookup method where a word is assigned its category according to which word-list it belongs to. This, however, is clearly an oversimplification. For instance, the word "Black" is a frequent surname and at the same time it is a frequent common word, thus the lexical information is not very reliable in this case. But by employing local context one can more robustly predict that in the context "Black described.." this word acts as a proper name and in the context "Black umbrella.." this word acts as a common word.

It is almost impossible to robustly estimate contexts larger than single focal word using word-based methods – even bigrams of words are too sparse. For instance, there are more than 50,000 distinct words in the Brown Corpus, thus there are $2^{50,000}$ potential word bigrams, but only a tiny fraction of them can be observed in the corpus. This is why words are often grouped into semantic classes. This, however, requires large manual effort, is not scalable and still covers only a fraction of the lexica. *Syntactic* context is much easier to estimate because the number of syntactic categories is much smaller than the number of distinct words.

A standard way to identify syntactic categories for word-tokens is part-of-speech (POS) tagging. There syntactic categories are represented as POS tags e.g. `NNS` - plural noun, `VBD` - verb past form, `JJR` - comparative adjective, etc. There exist several tag-sets which are currently in use – some of them reflect only the major syntactic information such as part-of-speech, number, tense, etc., whereas others reflect more refined information such as verb subcategorization, distinction between mass and plural nouns, etc.

Depending on the level of detail one tag-set can incorporate a few dozen tags where another can incorporate a few hundred, but still such tags will be considerably less sparse than individual words. For instance, there are only about 40 POS tags in the Penn Treebank tag-set, therefore there are only 2^{40} potential POS bigrams. Of course, not every word combination and POS tag combination is possible, but these numbers give a rough estimation of the magnitude of required data for observing necessary contexts for words and POS tags. This is why the "lexical lookup" method is the major source of information for word-based methods.

The "lexical lookup" method for deciding whether a capitalized word in a position where capitalization is expected (e.g. after a fullstop) is a proper name or a common word gives about an 8% error rate on the Brown Corpus. We developed and trained a POS tagger which reduced this error more than by half – achieving just above a 3% error rate. On the WSJ corpus the POS tagging advantage was even greater: our tagger reduced the error rate from 15% of the lexical lookup approach to 5%. This suggests that the error rate of a sentence splitter can be reduced proportionally by using the POS tagging methodology to predict whether a capitalized word after a period is a proper name or a common word.

1.2 The SATZ System

(Palmer and Hearst, 1997) described an approach which recognized the potential of the local syntactic context for the SBD problem. Their system, SATZ, used POS information for words in the local context of potential sentence splitting punctuation. However, what is interesting is that they found difficulty in applying a standard POS tagging framework for determining POS information for the words: "However, requiring a single part-of-speech assignment for each word introduces a processing circularity: because most part-of-speech taggers require predetermined sentence boundaries, the boundary disambiguation must be done before tagging. But if the disambiguations done before tagging, no part-of-speech assignments are available for the boundary determination system".

Instead, they applied a simplified method. The SATZ system mapped Penn Treebank POS tags into a set of 18 generic POS categories such as noun, article, verb, proper noun, preposition, etc. Each word was replaced with a set of these generic categories that it can take on. Such sets of generic syntactic categories for three tokens before and three tokens after the period constituted a context which was then fed into two kinds of classifiers (decision trees and neural networks) to make the predictions.

This system demonstrated reasonable accuracy (1.0% error rate on the WSJ corpus) and also exhibited robustness and portability when applied to other domains and languages. However, the N-grams of syntactic category sets have two important disadvantages in comparison to the traditional POS tagging which is usually largely based (directly or indirectly) on the N-grams of POS tags. First, syntactic category sets are much sparser than syntactic categories (POS tags) and, thus, require more data for training. Second, in the N-grams-only method

```
...<W C='RB' A='N'>soon</W><W C='.'>.</W> <W A='Y' C='NNP'>Mr</W><W C='A'>.</W>...

...<W C='VBD'>said</W> <W C='NNP' A='Y'>Mr</W><W C='A'>.</W>  <W C='NNP'>Brown</W>...

...<W C=','>,</W> <W C='NNP' A='Y'>Tex</W><W C='*'>.</W> <W C='DT'>The</W>...
```

Figure 1: Example of tokenization and markup. Text is tokenized into tokens represented as XML elements with attributes: `A='Y'` - abbreviation, `A='N'`- not abbreviation, `C` – part-of-speech tag attribute, `C='.'` - fullstop, `C='A'` - part of abbreviation, `C='*'` - a fullstop and part of abbreviation at the same time.

no influence from the words outside the N-grams can be traced, thus, one has to adopt N-grams of sufficient length which in its turn leads either to sparse contexts or otherwise to sub-optimal discrimination. The SATZ system adopted N-grams of length six. In contrast to this, POS taggers can capture influence of the words beyond an immediate N-gram and, thus, usually operate with N-grams of length two (bigrams) or three (three-grams). Furthermore, in the POS tagging field there exist standard methods to cope with N-gram sparseness and unknown words. Also there have been developed methods for *unsupervised* training for some classes of POS taggers.

1.3 This Paper

In this paper we report on the integration of the sentence boundary disambiguation functionality into the POS tagging framework. We show that sentence splitting can be handled *during* POS tagging and the above mentioned "circularity" can be tackled by using a non-traditional tokenization and markup conventions for the periods. We also investigate reducing the importance of pre-existing abbreviation lists and describe guessing strategies for unknown abbreviations.

2 New Handling of Periods

In the traditional Treebank schema, abbreviations are tokenized *together* with their trailing periods and, thus, stand-alone periods unambiguously signal end-of-sentence. For handling the SBD task we suggest tokenizing periods *separately* from their abbreviations and treating a period as an ambiguous token which can be marked as a fullstop (`'.'`), part-of-abbreviation (`'A'`) or both (`'*'`). An example of such markup is displayed on Figure 1. Such markup allows us to treat the period similarly to all other words in the text: a word can potentially take on one of a several POS tags and the job of a tagger is to resolve this ambiguity.

In our experiments we used the Brown Corpus and the Wall Street Journal corpus both taken from the Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993). We converted both these corpora from the original format to our XML format (as displayed on Figure 1), split the final periods from the abbreviations and assigned them with `C='A'` and `C='*'` tags according to whether or not the abbreviation was the last token in a sentence. There were also quite a few infelicities in the original tokenization and tagging of the Brown Corpus which we corrected by hand.

Using such markup it is straightforward to train a POS tagger which also disambiguates sentence boundaries. There is, however, one difference in the implementation of such tagger. Normally, a POS tagger operates on a text-span which forms a sentence and this requires performing the SBD before tagging. However, we see no good reason why such a text-span should necessarily be a sentence, because almost all the taggers do not attempt to parse a sentence and operate only in the local window of two to three tokens.

The only reason why the taggers traditionally operate on the sentence level is because there exists a technical issue of handling long text spans. Sentence length of 30-40 tokens seems to be a reasonable limit and, thus, having sentences pre-chunked before tagging simplifies life. This issue, however, can be also addressed by breaking the text into short text-spans at positions where the previous tagging history does not affect current decisions. For instance, a bigram tagger operates within a window of two tokens, and thus a sequence of word-tokens can be terminated at an unambiguous word because this unambiguous word token will be the only history used in tagging of the next token. A trigram tagger operates within a window of three tokens, and thus a sequence of word-tokens can be terminated when two unambiguous words follow each other.

3 Tagging Experiment

Using the modified treebank we trained a tri-gram POS tagger (Mikheev, 1997) based on a combination of Hidden Markov Models (HMM) and Maximum Entropy (ME) technologies. Words were clustered into ambiguity classes (Kupiec, 1992) according to sets of POS tags they can take on. This is a standard technique that was also adopted by the SATZ system[1]. The tagger predictions were based on the ambiguity class of the current word together with

[1]The SATZ system operated with a reduced set of 18 generic categories instead of 40 POS tags of the Penn Treebank tag-set.

Table 1: POS Tagging on sentence splitting punctuation and ambiguously capitalized words

Tagger Feature Set	*Error on Sentence Punct.*		*Error on Words in Mandatory Pos.*	
	Brown Corpus	WSJ Corpus	Brown Corpus	WSJ Corpus
Upper Bound	0.01	0.13	–	–
POS Tagger	0.25%	0.39%	3.15%	4.72%
POS Tagger Enhanced	0.20%	0.31%	1.87%	3.22%
POS Tagger/ No abbr. list	0.98%	1.95%	3.19%	5.29%
POS Tagger Enhanced/ No abbr. list	0.65%	1.39%	1.91%	3.28%

the POS trigrams: hypothesized current POS tag and partially disambiguated POS tags of two previous word-tokens. We also collected a list of abbreviations as explained later in this paper and used the information about whether a word is an abbreviation, ordinary word or potential abbreviation (i.e. a word which could not be robustly classified in the first two categories). This tagger employed Maximum Entropy models for tag transition and emission estimates and Viterbi algorithm (Viterbi, 1967) for the optimal path search.

Using the forward-backward algorithm (Baum, 1972) we trained our tagger in the *unsupervised* mode i.e. without using the annotation available in the Brown Corpus and the WSJ. For evaluation purposes we trained our tagger on the Brown Corpus and applied it to the WSJ corpus and vice versa. We preferred this method to ten-fold cross-validation because this allowed us to produce only two tagging models instead of twenty and also this allowed us to test the tagger in harsher conditions when it is applied to texts which are very distant from the ones it was trained on.

In this research we concentrated on measuring the performance only on two categories of word-tokens: on periods and other sentence-ending punctuation and on word-tokens in mandatory positions. Mandatory positions are positions which might require a word to be capitalized e.g. after a period, quotes, brackets, in all-capitalized titles, etc. At the evaluation we considered proper nouns (NNP), plural proper nouns (NNPS) and proper adjectives[2] (JJP) to signal a proper name, all all other categories were considered to signal a common word or punctuation. We also did not consider as an error the mismatch between "." and "*" categories because both of them signal that a period denotes the end of sentence and the difference between them is only whether this period follows an abbreviation or a regular word.

In all our experiments we treated embedded sentence boundaries in the same way as normal sentence boundaries. The embedded sentence boundary occurs when there is a sentence inside a sentence. This can be a quoted direct speech sub-sentence inside a sentence, this can be a sub-sentence embedded in brackets, etc. We considered closing punctuation of such sentences equal to closing punctuation of ordinary sentences.

There are two types of error the tagger can make when disambiguating sentence boundaries. The first one comes from errors made by the tagger in identifying proper names and abbreviations. The second one comes from the limitation of the POS tagging approach to the SBD task. This is when an abbreviation is followed by a proper name: POS information normally is not sufficient to disambiguate such cases and the tagger opted to resolve all such cases as "not sentence boundary". There are about 5-7% of such cases in the Brown Corpus and the WSJ and the majority of them, indeed, do not signal a sentence boundary.

We can estimate the upper bound for our approach by pretending that the tagger was able to identify all abbreviations and proper names with perfect accuracy. We can simulate this by using the information available in the treebank. It turned out that the tagger marked all the cases when an abbreviation is followed by a proper name, punctuation, non-capitalized word or a number as "not sentence boundary". All other periods were marked as sentence-terminal. This produced 0.01% error rate on the Brown Corpus and 0.13% error rate on the WSJ as displayed in the first row of Table 1.

In practice, however, we cannot expect the tagger to be 100% correct and the second row of Table 1 displays the actual results of applying our POS tagger to the Brown Corpus and the WSJ. General tagging performance on both our corpora was a bit better than a 4% error rate which is in line with the standard performance of POS taggers reported on these two corpora. On the capitalized words in mandatory positions the tagger achieved a 3.1-4.7% error rate which is an improvement over the lexical lookup approach by 2-3 times. On the sentence breaking punctuation the tagger performed extremely well – an error rate of 0.39% on the WSJ and 0.25% on the Brown Corpus. If we compare these results with the upper bound we see that the errors made by the tagger on the capitalized words and abbreviations

[2]These are adjectives like "American" which are always written capitalized. We identified and marked them in the WSJ and Brown Corpus.

instigated about a 0.25% error rate on the sentence boundaries.

We also applied our tagger to single-case texts. We converted the WSJ and the Brown Corpus to upper-case only. In contrast to the mixed case texts where capitalization together with the syntactic information provided very reliable evidence, syntactic information without capitalization is not sufficient to disambiguate sentence boundaries. For the majority of POS tags there is no clear preference as to whether they are used as sentence starting or sentence internal. To minimize the error rate on single case texts, our tagger adopted a strategy to mark all periods which follow abbreviations as "non-sentence boundaries". This gave a 1.98% error rate on the WSJ and a 0.51% error rate on the Brown Corpus. These results are in line with the results reported for the SATZ system on single case texts.

4 Enhanced Feature Set

(Mikheev, 1999) described a new approach to the disambiguation of capitalized words in mandatory positions. Unlike POS tagging, this approach does not use local syntactic context, but rather it applies the so-called *document-centered approach.* The essence of the document-centered approach is to scan the entire document for the contexts where the words in question are used unambiguously. Such contexts give the grounds for resolving ambiguous contexts.

For instance, for the disambiguation of capitalized words in mandatory positions the above reasoning can be crudely summarized as follows: if we detect that a word has been used capitalized in an unambiguous context (not in a mandatory position), this increases the chances for this word to act as a proper name in mandatory positions in the same document. And, conversely, if a word is seen only lowercased, this increases the chances to downcase it in mandatory positions of the same document. By collecting sequences and unigrams of unambiguously capitalized and lowercased words in the document and imposing special ordering of their applications (Mikheev, 1999) reports that the document-centered approach achieved a 0.4-0.7% error rate with coverage of about 90% on the disambiguation of capitalized words in mandatory positions.

We decided to combine this approach with our POS tagging system in the hope of achieving better accuracy on capitalized words after the periods and therefore improving the accuracy of sentence splitting. Although the document-centered approach to capitalized words proved to be more accurate than POS tagging, the two approaches are complimentary to each other since they use different types of information. Thus, the hybrid system can bring at least two advantages. First, unassigned by the document-centered approach 10% of the ambiguously capitalized words can be assigned using a standard POS tagging method based on the local syntactic context. Second, the local context can correct some of the errors made by the document-centered approach. To implement this hybrid approach we incorporated the assignments made by the document-centered approach to the words in mandatory positions to our POS tagging model by simple linear interpolation.

The third row of Table 1 displays the results of the application of the extended tagging model. We see an improvement on proper name recognition by about 1.5%: overall error rate of 1.87% on the Brown Corpus and overall error rate 3.22% on the WSJ. This in its turn allowed for better tagging of sentence boundaries : a 0.20% error rate on the Brown Corpus and a 0.31% error rate on the WSJ, which corresponds to about 20% cut in the error rate in comparision to the standard POS tagging.

5 Handling of Abbreviations

Information about whether a word is an abbreviation or not is absolutely crucial for sentence splitting. Unfortunately, abbreviations do not form a closed set, i.e., one cannot list all possible abbreviations. It gets even worse – abbreviations can coincide with ordinary words, i.e., "in" can denote an abbreviation for "inches", "no" can denote an abbreviation for "number", "bus" can denote an abbreviation for "business", etc.

Obviously, a practical sentence splitter which in our case is a POS tagger, requires a module that can guess unknown abbreviations. First, such a module can apply a well-known heuristic that single-word abbreviations are short and normally do not include vowels (Mr., Dr., kg.). Thus a word without vowels can be guessed to be an abbreviation unless it is written in all capital letters which can be an acronym (e.g. RPC). A span of single letters, separated by periods forms an abbreviation too (e.g. Y.M.C.A.). Other words shorter than four characters and unknown words shorter than five characters should be treated as *potential* abbreviations. Although these heuristics are accurate they manage to identify only about 60% of all abbreviations in the text which translates at 40% error rate as shown in the first row of Table 2.

These surface-guessing heuristics can be supplemented with the *document-centered approach* (DCA) to abbreviation guessing, which we call Positional Guessing Strategy (PGS). Although a short word which is followed by a period can potentially be an abbreviation, the same word when occurring in the same document in a different context can be unambiguously classified as an ordinary word if it is used without a trailing period, or it can be unambiguously classified as an abbreviation if it is used with a

Table 2: Error rate for different abbreviation identification methods

Corpus	*WSJ*	*Brown*
surface guess	39%	40%
surface guess and DCA	8.4%	9.5%
surface guess and DCA and abbr. list	0.8%	1.2%

trailing period and is followed by a lowercased word or a comma. This allows us to assign such words accordingly even in ambiguous contexts of the same document, i.e., when they are followed by a period.

For instance, the word "Kong" followed by a period and then by a capitalized word cannot be safely classified as a regular word (non-abbreviation) and therefore it is a potential abbreviation. But if in the same document we detect a context *"lived in Hong Kong in 1993"* this indicates that "Kong" is normally written without a trailing period and hence is not an abbreviation. Having established that, we can apply this findings to the non-evident contexts and classify "Kong" as a regular word (non-abbreviation) throughout the document. However, if we detect a context such as *"Kong., said"* this indicates that in this document "'Kong" is normally written with a trailing period and hence is an abbreviation. This gives us grounds to classify "Kong" as an abbreviation in all its occurrences within the same document.

The positional guessing strategy relies on the assumption that there is a consistency of writing within the same document. Different authors can write "Mr" or "Dr" with or without trailing period but we assume that the same author (the author of a document) will write consistently. However, there can occur a situation when a potential abbreviation is used as a regular word and as an abbreviation within the same document. This is usually the case when an abbreviation coincides with a regular word e.g. "Sun." (meaning Sunday) and "Sun" (the name of a newspaper). To tackle this problem, our strategy is to collect not only unigrams of potential abbreviations in unambiguous contexts as explained earlier but also their bigrams with the preceding word. Now the positional guessing strategy can assign ambiguous instances on the basis of the bigrams it collected from the document.

For instance, if in a document the system found a context *"vitamin C is"* it stores the bigram "vitamin C" and the unigram "C" with the information that it is a regular word. If in the same document the system also detects a context *"John C. later said"* it stores the bigram "John C." and the unigram "C" with the information that it is an abbreviation. Here we have conflicting information for the word "C" - it was detected as acting as a regular word and as an abbreviation within the same document – so there is not enough information to resolve ambiguous cases purely using the unigram. However, some cases can be resolved on the basis of the bigrams e.g. the system will assign "C" as an abbreviation in an ambiguous context *"... John C. Research ..."* and it will assign "C" as a regular word (non-abbreviation) in an ambiguous context *"... vitamin C. Research ..."*.

When neither unigrams nor bigrams can help to resolve an ambiguous context for a potential abbreviation, the system decides in favor of the more frequent category deduced from the current document for this potential abbreviation. Thus if the word "In" was detected as acting as a non-abbreviation (preposition) five times in the current document and two times as abbreviation (for the state Indiana), in a context where neither of the bigrams collected from the document can be applied, "In" is assigned as a regular word (non-abbreviation). The last resort strategy is to assign all non-resolved cases as non-abbreviations.

Apart from the ability of finding abbreviations beyond the scope of the surface guessing heuristics, the document-centered approach also allows for the classification of some potential abbreviations as ordinary words, thus reducing the ambiguity for the sentence splitting module. The second row of Table 2 shows the results when we supplemented the surface guessing heuristics with the document-centered approach. This alone gave a huge improvement over the surface guessing heuristics.

Using our abbreviation guessing module and an unlabeled corpus from New York Times 1996 of 300,000 words, we compiled a list of 270 abbreviations which we then used in our tagging experiments together with the guessing module. In this list we included abbreviations which were identified by our guesser and which had a frequency of five or greater. When we combined the guessing module together with the induced abbreviation list and applied it to the Brown Corpus and the WSJ we measured about 1% error rate on the identification of abbreviation as can be seen in the third row of Table 2.

We also tested our POS tagger and the extended tagging model in conjunction with the abbreviation guesser only, when the system was not equipped with the list of abbreviations. The error rate on capitalized words went just a bit higher while the error rate on the sentence boundaries increased by two-three times but still stayed reasonable. In terms of absolute numbers, the tagger achieved a 0.98% error rate on the Brown Corpus and a 1.95% error rate on the WSJ when disambiguating sentence boundaries. The extended system without the abbreviation list was about 30% more accurate and achieved a 0.65% error rate on sentence splitting on the Brown Corpus and 1.39% on the WSJ corpus as shown in the last row of Table 1. The larger impact on the WSJ corpus can be explained by the fact that it has a higher proportion of abbreviations than the Brown Corpus. In the Brown Corpus, 8% of potential sentence boundaries come after abbreviations. The WSJ is richer in abbreviations and 17% of potential sentence boundaries come after abbreviations. Thus, unidentified abbreviations had a higher impact on the error rate in the WSJ.

6 Conclusion

In this paper we presented an approach which treats the sentence boundary disambiguation problem as part of POS tagging. In its "vanilla" version the system performed above the results recently quoted in the literature for the SBD task. When we combined the "vanilla" model with the *document-centered approach* to proper name handling we measured about a 20% further improvement in the performance on sentence splitting and about a 40% improvement on capitalized word assignment.

POS tagging approach to sentence splitting produces models which are highly portable across different corpora: POS categories are much more frequent than individual words and less affected by unseen words. This differentiates our approach from word-based sentence splitters. In contrast to (Palmer and Hearst, 1997), which also used POS categories as predictive features, we relied on a proper POS tagging technology, rather than a shortcut to POS tag estimation. This ensured higher accuracy of the POS tagging method which cut the error rate of the SATZ system by 69%. On the other hand because of its simplicity the SATZ approach is probably easier to implement and faster to train than a POS tagger.

On single-case texts the syntactic approach did not show a considerable advantage to the word-based methods: all periods which followed abbreviations were assigned as "sentence internal" and the results achieved by our system on the single-case texts were in line with that of the other systems.

The abbreviation guessing module which combines the surface guessing heuristics with the document centered approach makes our system very robust to new domains. The system demonstrated strong performance even without being equipped with a list of known abbreviations which, to our knowledge, none of previously described SBD systems could achieve.

Another important advantage of our approach we see is that it requires potentially a smaller amount of training data and this training data does not need to be labeled in any way. In training a conventional sentence splitter one usually collects periods with the surrounding context and these samples have to be manually labeled. In our case a POS tagging model is trained on all available words, so syntactic dependencies between words which can appear in a local context of a period can be established from other parts of the text. Our system does not require annotated data for training and can be *unsupervisedly* trained from raw texts of approximately 300,000 words or more.

There are ways for further improvement of the performance of our system by combining it with a word-based system which encodes specific behavior for individual words. This is similar to how the SATZ system was combined with the Alembic system. This addresses the limitation of our syntactic approach in treating cases when an abbreviation is followed by a proper name always as "non sentence boundary". In fact we encoded one simple rule that an abbreviation which stands for an American state (e.g. Ala. or Kan.) always is sentence terminal if followed by a proper name. This reduced the error rate on the WSJ from 0.31% to 0.25%. Another avenue for further development is to extend the system to other languages.

References

Aberdeen, J., J Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. 1995. Mitre: Description of the alembic system used for muc-6. In *The Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland. Morgan Kaufmann.

Baum, L.E. 1972. An inequality and associated maximization techique in statistical estimation for probabilistic functions of a Markov process. *Inequalities* 3 (1972) 1-8.

Kupiec, Julian. 1992. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*.

Marcus, Mitchell, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–329.

Mikheev, A. 1999. A knowledge-free method for capitalized word disambiguation. In *Proceedings of the 37th Conference of the Association for Computational Linguistics (ACL'99)*, pages 159–168. University of Maryland.

Mikheev, A., 1997. *LT POS – the LTG part of speech tagger*. Language Technology Group, University of Edinburgh. www.ltg.ed.ac.uk/software/pos.

Palmer, D. D. and M. A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics.*

Reynar, J. C. and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing (ANLP'97),* Washington, D.C.

Riley, M.D. 1989. Some applications of tree-based modeling to speech and language indexing. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 339–352. Morgan Kaufman.

Viterbi, A.J. 1967. Error bounds for convolutional codes and an asymptomatically optimal decoding algorithm. *IEEE Transactions on Information Theory.*