Unsupervised Learning of Period Disambiguation for Tokenisation

Helmut Schmid
IMS-CL, University of Stuttgart
Azenbergstr. 12, D-70174 Stuttgart, Germany
Email: schmid@ims.uni-stuttgart.de

April 5, 2000

Abstract

A language-independent period disambiguation method is presented which achieves high accuracy (> 99.5 %) and requires no other information than the corpus which is to be tokenised. The presented method automatically extracts statistical information about likely abbreviations, about sentence-initial words and about words which precede or follow numbers. This information is used to disambiguate periods and to recognise ordinal numbers and abbreviations. The recognition of abbreviations in languages with large compound nouns like German is enhanced by suffix analysis.

1 Introduction

Tokenisation of text corpora is – at least for alphabetic languages – often considered trivial. In fact, a simple program which replaces whitespace with word boundaries and cuts off leading and trailing quotation marks, parentheses and punctuation already produces fairly good results. There is only one major problem: periods are not adequately dealt with. A period at the end of a token is either part of the token in case of ordinal numbers (as in *2. International Congress*) or of an abbreviation (as in *Mrs. Jones*), or it is a full stop which marks the end of a sentence, or it is both (as in the sentence *The primary was won by Ivan Allen Jr.*).

The following heuristic is often used to disambiguate periods:

1. Leave the period attached

   - if it is immediately followed by an alphabetic or numeric character (i.e. the period is token-internal as in *www.uni-stuttgart.de* or in *12.07*)
   - if a known abbreviation precedes it as in *Mr.* or *etc.*
   - if a lower case character or a punctuation character like *, ; : ? !* follows it as in *etc.,* or in *Mr. and Mrs. Jones*

2. Cut off periods in all other cases.

This heuristic correctly disambiguates most periods. However, some problems are still unsolved: Numbers with a trailing period are always classified as cardinal numbers (cmp. the counter-example *2. International Congress*) and sentence boundaries after abbreviations are not detected. The most serious problem, however, is that the list of abbreviations is incomplete because new abbreviations appear in almost every corpus. Therefore the list of abbreviations has to be adapted to each corpus. Otherwise tokenisation results might be poor. So, there is a need for a tokeniser which automatically adapts to new corpora and achieves high accuracy with arbitrary types of text. Such a tokeniser will be presented in this paper.

Several approaches were developed for period disambiguation in the past. Palmer and Hearst [Palmer and Hearst, 1994] obtained 98.5 % accuracy on Wall Street Journal data with a neural network. Michael Riley [Riley, 1989] reported 99.8 % accuracy on the Brown corpus for his decision tree approach. Reynar and Ratnaparkhi [Reynar and Ratnaparkhi, 1997] achieved 98.8 % accuracy on Wall Street Journal data and 97.9 % on the Brown corpus with maximum entropy modelling. A similar approach with slightly better results was presented in [Mikheev, 1998].

In contrast to these methods, Grefenstette and Tapanainen's semi-automatically trained disambiguation method [Grefenstette and Tapanainen, 1994] requires no hand-annotated training data. They reported 99.07 % correctly disambiguated sentence boundaries for the Brown corpus with a system which had access to a lexicon and a list of frequent abbreviations. Andrei Mikheev recently presented a part-of-speech tagger based approach to sentence boundary detection and reports up to 99.80 % accuracy on the Brown corpus and 99.69 % on Wall Street Journal data [Mikheev, 2000]. His system is also trained on unlabelled data.

This paper describes a statistical period disambiguation method which requires neither disambiguated training data nor lexica or any other information apart from the corpus which is to be tokenised. In a first pass over the corpus, the tokeniser extracts statistical information about likely abbreviations and names, about lower-case words (which are good indicators for sentence boundaries when they appear in capitalised form after a period), and about the frequency of words before and after numbers (needed for number disambiguation). In the second pass, the extracted information is used to disambiguate the periods in the corpus. The presented method was evaluated on the Brown corpus and the Wall Street Journal corpus.

The paper is organised as follows. Section 2 describes the disambiguation method in detail. The evaluation results are presented in section 3 and compared with other methods in section 4. Section 5 concludes with a summary.

2 Period Disambiguation

We distinguish four types of period occurrences which require disambiguation, namely:[1]

1. **w. w'** The period is preceded by a non-numeric token and followed by a token which starts with a lower-case character like *"Mr. and"*.
2. **w. W'** The period is preceded by a non-numeric token and followed by a token which starts with something else than a lower-case character like *"Mr. Jones"* or *"etc.)"* or *"Okla. The"*.
3. **z. w'** The period is preceded by a number and followed by a token which starts with a lower-case character like *"1. of"*.
4. **z. W'** The period is preceded by a number and followed by a token which does not start with a lower-case character like *"2. Congress"* of *"1. He"*.

The first case turned out to be more ambiguous than expected. On the one hand, there were some words – like *amnesty international* in our German newspaper corpus – which were always written in lower case even at the beginning of a sentence. On the other hand, articles in our corpus were often followed by a short sequence of lower-case letters which indicated the author of the article, so that the last sentence of the article was followed by a lower-case token.

Disambiguation methods for the four cases are described in the following sections.

[1]We will use `w` and `w'` for lower-case words and `W` and `W'` for capitalised words. If both, `w` and `W` appear in a formula, then `W` is the capitalised variant of `w`. The same holds for `w'` and `W'`.

2.1 Disambiguation of w. W'

We start with the second and most frequent case where the period is followed by a token which does not start with a lower-case character. There are three possibilities. Either the period belongs to an abbreviation or the period is a full stop or it is both. If the sentence ends after the period and the next word starts with an upper-case character, then two sub-cases have to be distinguished: the next word is either always capitalised (like the word *Jones*) or only at the beginning of a sentence (like the article *The*). If we want the tokeniser to "de-capitalise" the next word in the latter case, then 5 possible outputs exist:

1. **w. W'** abbreviation + capitalised word as in *"Mr. Jones"*
2. **w . W'** normal token + full stop + capitalised word as in *"Department . Richard"*
3. **w . w'** normal word + full stop + de-capitalised word as in *"cruelty . the"*
4. **w. . W'** abbreviation + full stop + capitalised word as in *"Rd. . Henry"*
5. **w. . w'** abbreviation + full stop + de-capitalised word as in *"p.m. . the"*

2.1.1 The Probability Model

In order to disambiguate between the five alternatives, we compute their probabilities and then choose the most likely one. It will be convenient to divide the probabilities by the term $P(w\ .)P(w')$. Assuming that words are independently distributed (e.g. $P(W'|w.) = P(W')$), we obtain the following probability ratios

$$\begin{aligned}
\frac{P(w.\ W')}{P(w\ .)\ P(w')} &\approx \frac{P(w.)}{P(w\ .)}\frac{P(W')}{P(w')} = A\ B \\
\frac{P(w\ .\ W')}{P(w\ .)\ P(w')} &\approx \frac{P(w\ .)}{P(w\ .)}\frac{P(W')}{P(w')} = B \\
\frac{P(w\ .\ w')}{P(w\ .)\ P(w')} &\approx \frac{P(w\ .)}{P(w\ .)}\frac{P(w')}{P(w')} = 1 \\
\frac{P(w.\ .\ W')}{P(w\ .)\ P(w')} &\approx \frac{P(w.)}{P(w\ .)}P(.)\frac{P(W')}{P(w')} = A\ B\ P(.) \\
\frac{P(w.\ .\ w')}{P(w\ .)\ P(w')} &\approx \frac{P(w.)}{P(w\ .)}P(.)\frac{P(w')}{P(w')} = A\ P(.)
\end{aligned}$$

where $A = P(w.)/P(w\ \ .)$ and $B = P(W')/P(w')$. The tokeniser chooses the analysis with the highest probability (ratio). The fourth expression is always smaller than the first, so it will never be chosen. The most frequent cases are:

- $A < 1, B < 1$: output `w . w'`
- $A < 1, B < 1$: output `w . W'`
- $A > 1, B > 1$: output is `w. W'`

If $A > 1$ and $B < 1$, then the most likely output is usually (but not always) `w. . w'`. It turns out that the results obtained with the following decision table are almost identical[2] to the results obtained when the four probabilites are actually computed and compared.

	B > 1	B < 1
A > 1	w. W'	w. . w'
A < 1	w . W'	w . w'

If we define an *abbreviation* to be a word $w.$ where $P(w.) > P(w\ \ .)$ (i.e. `w.` is more frequent than `w` at the end of a sentence) and further define that `w` is a *word* if $P(W) < P(w)$ (i.e. `w` is more frequent than `W`, then we obtain the following intuitive disambiguation algorithm:

```
If w. is an abbreviation then
    if w' is a word then
        print w. . w'
    else
        print w. W'
else
    if w' is a word then
        print w . w'
    else
        print w . W'
```

2.1.2 Parameter Estimation

So far, we have not explained how the probability ratios $P(w.)/P(w\ \ .)$ and $P(w)/P(W)$ are estimated from unlabelled training data. If we assume that the probability of a word is the same at the beginning of sentences and within sentences, then we can estimate $P(w)/P(W)$ by the frequency ratio $F(w)/F(W)$ where $F(w)$ is the frequency of the word `w` within sentences and $F(W)$ is the frequency of the capitalised variant `W` within sentences.

$P(w.)/P(w\ \ .)$ is more difficult to estimate because direct counting of `w.` and `w .` requires disambiguated training data. However, even in unlabelled data, there are cases which are almost unambiguous. If some potential abbreviation `w.` is

[2]Experimental accuracy was only 0.03 percent worse.

followed by a lower-case word or punctuation like `,;:?!`, then it is probably an abbreviation. On the other hand, if a potential abbreviation often appears without a trailing period, then is probably not an abbreviation. We will therefore estimate the fraction $P(w.)/P(w\ \ .)$ as follows:

$$\frac{P(w.)}{P(w\ \ .)} \approx \frac{P(w.\ l)}{P(w\ .\ L)} = \frac{P(w.\ l)}{P(w)\ P(.\ L \mid w)} \approx \frac{P(w.\ l)}{P(w)\ P(.\ L)} \approx \frac{F(w.\ l)}{F(w)\ P(.\ L)}$$

The letter `l` matches any lower-case word, the letter `L` matches any capitalised word which is normally written in lower case. $F(w.\ l)$ is the frequency of word w occurring before lower case words. The probability $P(.\ L)$ is easily estimated from the corpus, once we know which words are normally written in lower case.

There is an important class of abbreviations which are not discovered well enough by the presented method, namely titles. They typically occur before names which are capitalised. So, most occurrences of titles are ambiguous and not counted by this method. In order to detect titles, we need a different method. If `w.` is not an abbreviation and therefore the period is a full stop, then we expect that the ratio of "true" capitalised words (like *Jones*) and capitalised "lower case" words (like `This`) following `w.` is about the same as the overall ratio at sentence start. If the ratio actually found for word `w` deviates to much from this expectation, then it is likely to be an abbreviation.

$$\frac{P(w\ \ .\ \ U)}{P(w\ \ .\ \ L)} \approx \frac{P(.\ \ U)}{P(.\ \ L)}$$

`U` matches here any capitalised word form which rarely or never appears in lower-case.

Abbreviations which appear predominantly at the end of a sentence are still not detected, but they cause no major problems because they are so rare.

2.1.3 Suffix Analysis

In languages with a rich derivational morphology like German, there are many abbreviations with common endings. Typical examples are German street names like *Kaiserstr., Azenbergstr., Hölderlinstr.*. The individual street names are too rare to be reliably recognised as abbreviations. If we look at the endings, however, we will find that words ending in *str.* are likely to be abbreviations.

Therefore, we compute the probability ratio $P(w.)/P(w\ .)$ not only for words but also for word endings and a word `w` is considered an abbreviation if either the estimate of $P(w.)/P(w\ .) > 1$ or the estimate of $P(s.)/P(s\ .) > 1$, where `s` is the 3-, 4-, or 5-character suffix of `w`.

2.1.4 Smoothing

In some of the above formulas, we devide by a frequency term which could be zero. Therefore, we smooth frequencies by adding a small constant (0.1). A more sophisticated smoothing method is unlikely to improve the results very much because the probability estimates are only rough approximations.

2.2 Disambiguation of w. w'

Now, we turn to the disambiguation of periods which are followed by lower-case words. These periods are usually abbreviations, but there are a few exceptions.

- Some words are always written in lower case, even at the beginning of a sentence like *amnesty international*.
- Sentences may be followed by something which is not a sentence, e.g.
  - by an incomplete sentence in which the first half has been cut off (caused e.g. by an editing mistake)
  - by a list marker, e.g. a)
  - by some other material e.g. a short sequence of lower case letters indicating the author of the preceding article as in our German newspaper corpus

It is possible to automatically recognise tokens which appear in lower case at sentence start if they are frequent enough. To this end, we count how often a lower case token appears after an unambiguous full stop (i.e. where the preceding token is not an abbreviation and therefore $P(w\ .)$ is much larger than $P(w.)$). Because "normal" lower-case words sometimes appear after a full stop (e.g. because of editing mistakes), we have to compare the frequency after full stops with the overall frequency of the word. We classify a lower-case word `w` as one of those words which are written in lower case at the beginning of a sentence if

$$F(.\ w)\ >\ log((F(w) - F(.\ w) + 1)$$

This heuristic worked very well. It correctly recognised this type of lower-case tokens in our German corpus even when an identical and more frequent "normal" word existed without generating false positives.

The disambiguation rule for periods which are preceded by a non-numeric token `x` and followed by a token `y` is then:

If `x.` is an *abbreviation* or has an abbreviation suffix then

```
    if y is the capitalised form of the lower-case word w' then
        print x.  . w'
    else
        print x. y
else if y is a lower-case token or a punctuation character then
    if y is always written in lower-case then
        print x  . y
    else
        print x. y
else
    print x  . y
```

2.3 Disambiguation of z. W

A similar statistical disambiguation method was developed for periods occurring after numbers. These periods are either part of an ordinal number or they mark the end of the sentence or both. We ignore the last possibility for the moment and develop a disambiguation method for the first two cases.

2.3.1 The Probability Model

Disambiguation between ordinal and cardinal numbers requires more context than disambiguation of abbreviations because the word preceding the number often provides more information for the disambiguation than the number itself (cmp. *Schillerstraße 2.* (Schiller Street 2.) vs. *am 2.* (on the 2.)). Therefore, we compute the probability ratio

$$\frac{P(w\ \ z.\ W')}{P(w\ \ z\ \ .\ W')} \approx \frac{P(w\ \ z.)\,P(W' \mid z.)}{P(w\ \ z)\,P(. \mid z)\,P(W' \mid .)} = \frac{P(w\ \ z.)}{P(w\ \ z)}\,\frac{P(z.\ W')}{P(.\ W')}\,\frac{P(z)}{P(z.)}\,\frac{P(.)}{P(z\ \ .)}$$

We assume here that a token depends only on the preceding token (and not on the last but one). We make the further assumption that $\frac{P(.)}{P(z\ \ .)} \approx \frac{P(.\ L)}{P(z\ \ .\ L)}$. Again, we use the letter `L` to subsume any capitalised word which would be written with lower-case characters within a sentence.

$$\frac{P(w\ \ z.\ W')}{P(w\ \ z\ \ .\ W')} \approx \underbrace{\frac{P(w\ \ z.)}{P(w\ \ z)}}_{C}\ \underbrace{\frac{P(z.\ W')}{P(.\ W')}}_{D}\ \underbrace{\frac{P(z)}{P(z.)}}_{E}\ \underbrace{\frac{P(.\ L)}{P(z\ \ .\ L)}}_{F}$$

We classify a number as an ordinal number if the ratio is larger than 1. Otherwise it is a cardinal number at the end of a sentence.

2.3.2 Parameter Estimation

The factor F is estimated by dividing the frequency of `. L` by the frequency of `z . L`. We count each number which is followed by a period and a capitalised "lower-case" word as a cardinal number and ignore the less frequent case in which `z.` is an ordinal number at the end of a sentence.

In order to estimate $\underline{P(w\ z.)/P(w\ z)}$ (factor C above), we assume that $P(l|w\ z.) \approx P(l)$ (i.e. that the probability of lower-case words after ordinal numbers is the same as the overall probability of lower-case words). Then we get

$$\frac{P(w\ z.)}{P(w\ z)} = \frac{P(w\ z.\ l)}{P(w\ z)\ P(l|w\ z.)} \approx \frac{P(w\ z.\ l)}{P(w\ z)\ P(l)} \approx \frac{F(w\ z.\ l)}{F(w\ z)\ P(l)}$$

In other words, we count how often `w z.` appears before a lower-case word and devide by the frequency[3] of `w z` and the probability of lower-case words.

Factor D is estimated by the ratio $F(z)/F(z.)$. The frequencies $F(z)$ and $F(z.)$ are obtained by summing the frequencies $F(w\ z)$ and $F(w\ z.)$ (see above) for all words w.

Finally, we have to estimate factor D. If `x` matches any token such that `x.` is not a token, and if almost all tokens in sentence-final position are matched by `x`, then $P(.\ W')$ can be estimated as follows

$$P(.\ W') \approx P(x\ .\ W') \approx F(x\ .\ W')/N$$

It remains to estimate $P(z.\ W')$. This is difficult because `z. W'` is ambiguous: the period might be a full stop. Sometimes, the word which precedes `z` is a strong indicator in either direction, but we cannot rely on this information because for many `W'` such indicators might be missing. Instead we count how often `W'` appears after `z.` (without disambiguating the number) and compare the resulting frequency with the frequency which we would expect if `z` were always a cardinal number. The expected frequency is computed as follows

$$\begin{aligned} F_{exp}(z\ .\ W') &= P(z\ .\ W')\ N \approx P(z\ .)\ P(W'|.)\ N = \frac{P(z\ .\ L)}{P(L|z\ .)}\ P(W'|.)\ N \\ &\approx \frac{P(z\ .\ L)}{P(L|.)}\ P(W'|.)\ N \approx F(z\ .\ L)\ \frac{P(.\ W')}{P(.\ L)} \approx F(z\ .\ L)\ \frac{F(x\ .\ W')}{F(x\ .\ L)} \end{aligned}$$

If the observed frequency is significantly higher than the expected frequency, then we can use the difference between the observed and the expected frequency

[3]We simply use the frequency of `w z` within sentences. It would be more accurate to multiply this frequency by $1 - P(.)$ in order to account for possible occurrences of `w z` at the end of a sentences.

as an estimate for $F(z.\,W')$. Otherwise, $P(z.\,W')$ cannot be estimated and the fallback method below has to be used.

$$P(z.\,W') = \begin{cases} [F(z.\,W') - F_{exp}(z.\,W')]/N & \text{if } F(z.\,W') \text{ is significantly larger} \\ & \text{than expected} \\ \text{undefined} & \text{otherwise} \end{cases}$$

2.3.3 Fallback Strategy

If $P(w\ z.)/P(w\ z)$ cannot be estimated reliably because the frequencies $F(w\ z.)$ and $F(w\ z)$ are too small, then we ignore `w` and use the following ratio for disambiguation:

$$\frac{P(z.\,W')}{P(z\ .\,W')} \approx \frac{P(z.)}{P(z)}\,\frac{P(z.\ W')}{P(.\ W')}\,\frac{P(z)}{P(z.)}\,\frac{P(.\ L)}{P(z\ .\ L)} = \underbrace{\frac{P(z.\ W')}{P(.\ W')}}_{D}\ \underbrace{\frac{P(.\ L)}{P(z\ .\ L)}}_{F}$$

If $P(z.\,W')/P(.\,W')$ cannot be estimated, then we use the formula

$$\frac{P(w\ z.\ W')}{P(w\ z\ .\ W')} \approx \frac{P(w\ z.)}{P(w\ z)}\,\frac{P(W'|z.)}{P(.|z)\,P(W'|.)} \approx \underbrace{\frac{P(w\ z.)}{P(w\ z)}}_{C}\ \underbrace{\frac{P(W')}{P(w')+P(W')}}_{:=G}\ \frac{1}{P(.|z)}$$

where $P(w')$ is the probability of the lower-case variant of `W'` if `W'` is a capitalised token. Otherwise $P(w') := 0$.

Because we do not know the probability of `W'` after `z.`, we assume that it does not differ from the overall probability $P(W')$. We further assume that the probabilities of `W'` and `w'` after full stops are identical to their apriori probabilities. $P(W')$ and $P(w')$ are easily estimated by their within-sentence frequencies. In order to estimate $1/P(.|z)$, we use the approximation

$$\frac{1}{P(.|z)} = \frac{P(z)}{P(z\ .)} = \frac{P(z)\,P(L|z\ .)}{P(z\ .\ L)} \approx \frac{P(z)\,P(L|.)}{P(z\ .\ L)} \approx \frac{F(z)}{F(z\ .\ L)}\,P(L|.)$$

The fallback method for the computation of the probability ratio is therefore

$$\frac{P(w\ z.\ W')}{P(w\ z\ .\ W')} \approx \underbrace{\frac{P(w\ z.)}{P(w\ z)}}_{C}\ \underbrace{\frac{P(W')}{P(w')+P(W')}}_{G}\ \frac{F(z)}{F(z\ .\ L)}\,P(L|.)$$

2.3.4 Equivalence classes

There are so many numbers that the above parameters cannot be estimated for all of them. However, numbers fall into a few classes with similar distributional properties within the classes. Therefore, we define equivalence classes and estimate the parameters for the equivalence classes rather than the individual numbers. The following equivalence classes are used:

- **1, 2, ..., 12, 01, 02, ..., 09** These numbers could be days or months in dates.
- **13, 14, ..., 31** These numbers could be the days in dates.
- **1900, 1901, ..., 1999** These numbers could be the years[4] in dates.
- **32, 33, ..., 1899, 2000, 2001, ...**
- **0.00, 0.01, ..., 23.59** These numbers could be time specifications.
- numbers containing commata
- numbers containing periods (excepted those above)
- numbers containing colons (`:`)
- numbers containing slashes (`/`)
- numbers containing blanks like `600 501 01`
- all other numbers

Wherever a `z` appears in the disambiguation formulas of chapter **??**, it has to be replaced with its equivalence class `[z]`.

2.3.5 Sentence-Final Ordinal Numbers

So far, we have ignored that ordinal numbers may appear at the end of a sentence. In order to detect this case, we first compare the probability of the sentence-final ordinal number with the probability of the cardinal number. We assume that the probability of periods is the same after cardinal and after ordinal numbers. Then it is sufficient to look at the preceding word in order to disambiguate.

$$\frac{P(w\ z.\ .\ W')}{P(w\ z\ .\ W')} \approx \frac{P(w\ z.)}{P(w\ z)} \frac{P(.|z.)}{P(.|z)} \frac{P(W'|.)}{P(W'|.)} \approx \underbrace{\frac{P(w\ z.)}{P(w\ z)}}_{C}$$

[4]Sorry, this algorithm is not yet Y2K compatible.

Now we compare the probability of a sentence-internal and a sentence-final ordinal number. We assume that the probability of W' and w' is the same after periods and after ordinal numbers.

$$\frac{P(w\ z.\ .\ W')}{P(w\ z.\ W')} \approx \frac{P(w\ z.)}{P(w\ z.)}\frac{P(W'|.)}{P(W'|z.)}P(.|z.) \approx \underbrace{\frac{P(W') + P(w')}{P(W')}}_{1/G} P(.)$$

We only choose a sentence-final ordinal number if the next word `W'` is a strong indicator for a sentence boundary, i.e. if $P(w')$ is much larger than $P(W')$.

For the disambiguation of numbers, we use the following decision rule which outputs ordinal numbers in sentence-final position only when they are more likely than both alternatives.

```
if $P(w\ z.\ .\ W')/P(w\ z\ .\ W') > 1$ and $P(w\ z.\ .\ W')/P(w\ z.\ W') > 1$ then
    output w z. . W'
else
    if $P(w\ z.\ W')/P(w\ z\ .\ W') > 1$ then
        output w z. W'
    else
        output w z . W'
```

2.4 Disambiguation of z. w

All numbers which are followed by a period and a lower case token are classified as ordinal numbers without a sentence boundary. We ignore here the rare cases where a lower case token appears after a sentence boundary.

2.5 Further Improvements

The method presented so far worked very well for our German newspaper corpus. Because no manually disambiguated German test corpus was available for evaluation, the method was tested on the Penn Treebank version of the Brown corpus as described in section 3. The resulting error rate on this corpus was 0.9 %.

This is quite good compared to most other approaches. However, an analysis of the errors revealed that improvements were possible. 23 percent of the errors were caused by incorrectly disambiguated numbers. It turned out that periods after numbers are not ambiguous in the Brown corpus because they are always classified as sentence boundaries (similar in the Wall Street Journal corpus discussed later). Therefore, the presented disambiguation method for numbers turned out to be useless for English and it was replaced by a simple heuristic which cuts off periods after numbers unless the period is followed by a lower-case character or a punctuation character.

Another frequent error was incorrect disambiguation of single uppercase letter which are followed by a period. These tokens are ambiguous because they are either part of a name as in `Henry L. Bowden` or they are full stops as in `We obtain an equation in terms of H.` Therefore a name recogniser was developed based on an idea of Andrei Mikheev [Mikheev, 1999] which extracts likely names from the corpus during the first pass. The period disambiguator was augmented by a heuristic which classifies an uppercase token which ends with a period as an abbreviation (example: `John W. Ball`) if

- it is one of the recognised names or a sequence of the form letter + period + letter ... of length 2 or more (like `L.` or `C.J.`) and
- the token after the period is also a name.

unless

- the token is a name which typically occurs at the end of a name sequence (like `Co.`) and
- the next token is a name which typically occurs at the beginning of a name sequence (like `Mr.`)

in which case the token is classified as an abbreviation at a sentence boundary as in `‘‘...McDonald & Co.  Mr. Sheridan...’’` This method works only for languages like English where proper names are the only capitalised words but not e.g. for German.

Another class of abbreviations which were not recognised very well are abbreviations like `No.` which appear predominantly before numbers. In order to enhance the recognition of these abbreviations, a function was added which counts how often potential abbreviations appear before numbers ($F(w.\ z)$ where z matches any number) and compares it to the number of occurrences of the potential abbreviation at the end of a sentence ($F(w.\ L)$ where L matches any capitalised token which is more frequent when written in lower case). Tokens found by this method are classified as abbreviations if they appear before numbers.

The evaluation results reported in the next section were obtained with a version of the tokeniser which included these enhancements and the simplified method for the disambiguation of numbers.

3 Evaluation

The presented period disambiguation method was evaluated on the tagged version of the Brown corpus and the Wall Street Journal corpus which are part of the Penn Treebank corpus [Marcus et al., 1993]. A reference corpus for the period disambiguation task was obtained by discarding the part-of-speech and noun phrase information from this corpus. An untokenised version of the Brown corpus was restored from the reference corpus by removing blanks between punctuation and preceding words, removing full stops which are preceded by an abbreviation with a final period[5] and so on. Several tokenization errors in the reference corpus were corrected. 0.2 percent of the ambiguous periods were affected by these corrections.

After preparing the Brown corpus, the tokenizer was trained on the restored version. The Brown corpus was then retokenized. The result was compared to the reference corpus.

Overall, the Brown corpus contained 1164625 tokens and 53747 periods which had to be disambiguated. Token-internal periods like `14.2` or `U.N.-chartered` were not evaluated because they are unambiguous. The tokenizer made 162 mistakes which translate to an accuracy of 99.70 %. The accuracy wrt. sentence boundary detection was 99.79 %.[6]

54 errors were due to unrecognised abbreviations. The most frequent ones were `La.` (7 errors), `in.` (4 errors) and `Miss.` (3 errors). All of these were not recognised because of ambiguities. `La` e.g. appears in foreign names and expressions like `La Dolce Vita`.

27 errors were caused by errorneous insertion of a sentence boundary in names as in `Mr. . Average Citizen`. These errors occurred if the name after the period was not recognised during training and the lower case form of the name was frequent.

20 errors were caused by single upper-case letters which were part of enumerations. These enumeration items were often mistokenised as abbreviations with or without a following full stop, whereas the respective periods in the Penn Treebank version of the Brown corpus were cut off from the preceding upper case letter.

16 tokens (among them `hel`[7], `oui`, `Rak`, `I`, `ell`, `hon`, `Wow`, `Hap`, `lb` (2 times), `Mass`, `Pa` (4 times) and `fig`) were misclassified as abbreviations. 40 errors occurred because a sentence boundary after an abbreviation was not recognised.

[5]The period is duplicated in the Brown corpus when an abbreviation appears at the end of a sentence.

[6]Sentence boundary detection is easier because it is not an error when an abbreviation in sentence-final position is not recognised. Furthermore sentences ending with an exclamation mark or question mark increase the number of correctly recognised sentence boundaries.

[7]the intentionally misspelled word `hell`

The same evaluation was carried out on sections 2 through 6 of the Wall Street Journal corpus which were also used by Palmer and Hearst [Palmer and Hearst, 1994] and by Reynar and Ratnaparkhi [Reynar and Ratnaparkhi, 1997] and by Andrei Mikheev [Mikheev, 2000]. The whole WSJ corpus was used for training. Again, a couple of tokenisation errors had to be corrected. The resulting accuracy wrt. sentence boundary detection was 99.56 % (61 errors). The accuracy on the whole WSJ corpus was 99.62 %. Period disambiguation accuracy could not be determined for the WSJ corpus because abbreviations in sentence-final position are not marked in the WSJ.

4 Comparison

The results in [Grefenstette and Tapanainen, 1994] and in [Mikheev, 2000] can be directly compared to ours because their methods also require no labelled training data. Grefenstette and Tapanainen report 98.35 % accuracy for sentence boundary detection in the Brown corpus when no lexicon and no precompiled list of abbreviations was used (99.07 % with these additional sources of information). Andrei Mikheev reports 99.80 % accuracy on the Brown corpus and 99.69 % on WSJ data. These are the best results we know of. Our method achieves almost the same accuracy as Mikheev's on the Brown corpus, but 30 percent lower accuracy on WSJ data. In contrast to Mikheev's method which requires a lexicon with part-of-speech information for the target language, our method is applicable to any language using the Latin alphabet without modification[8].

5 Summary

A period disambiguation method was presented which achieved high accuracy without prior knowledge of the corpus, the domain or the language. The presented method extracts a list of likely abbreviations, a list of likely names, a list of lower-case words, statistics about the preceding and following words of numbers and other information from the raw corpus data by statistical means. The extracted information is used to tokenise the corpus in a second pass. The accuracy of the presented method was as good or better than all but one of the other methods we know of, and none of them matched it wrt. adaptability to other languages.

[8]It might be useful to turn on or off some of the mentioned heuristics in order to get optimal results.

References

[Grefenstette and Tapanainen, 1994] Grefenstette, G. and Tapanainen, P. (1994). What is a word, what is a sentence? problems of tokenization. In *Proceedings of the 3rd Conference on Computational Lexicography and Text Research (COMPLEX'94)*, Budapest.

[Marcus et al., 1993] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[Mikheev, 1998] Mikheev, A. (1998). Feature lattices for maximum entropy modelling. In *acl98*, pages 845–848, Montreal, Canada.

[Mikheev, 1999] Mikheev, A. (1999). A knowledge-free method for capitalized word disambiguation. In *Proceedings of the 37th Annual Meeting of the ACL, University of California*, pages 159–166, Maryland.

[Mikheev, 2000] Mikheev, A. (2000). Tagging sentence boundaries. University of Edinburgh.

[Palmer and Hearst, 1994] Palmer, D. D. and Hearst, M. A. (1994). Adaptive sentence boundary disambiguation. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, Stuttgart, Germany.

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

[Riley, 1989] Riley, M. D. (1989). Some applications of tree-based modelling to speech and language indexing. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 339–352. Morgan Kaufmann.