A Comparative Evaluation of a New Unsupervised Sentence Boundary Detection Approach on Documents in English and Portuguese

Jan Strunk[1], Carlos N. Silla Jr.[2], and Celso A. A. Kaestner[2]

[1] Sprachwissenschaftliches Institut, Ruhr-Universität Bochum,
44780 Bochum, Germany
strunk@linguistics.rub.de
[2] Pontifical Catholic University of Paraná,
Rua Imaculada Conceição 1155, 80215-901 Curitiba, Brazil
{silla,kaestner}@ppgia.pucpr.br

Abstract. In this paper, we describe a new unsupervised sentence boundary detection system and present a comparative study evaluating its performance against different systems found in the literature that have been used to perform the task of automatic text segmentation into sentences for English and Portuguese documents. The results achieved by this new approach were as good as those of the previous systems, especially considering that the method does not require any additional training resources.

1 Introduction

We are living today in an era of information overload. The web alone contains about 170 terabytes of information, which is roughly 17 times the size of the printed material in the Library of Congress of the USA; cf. [1]. However, it is becoming more and more difficult to use the available information. Many problems such as the retrieval and extraction of information and the automatic summarization of texts have become important research topics in computer science. The use of automatic tools for the treatment of information has become essential to the user because without those tools it is virtually impossible to exploit all the relevant information available on the Web.

One pre-processing component that is essential to most text-based systems is the automatic segmentation of a text into sentences. Existing systems for sentence boundary detection mostly either use a set of heuristics or a supervised machine learning approach. The drawback of both these approaches is that adapting them to new languages can be time and resource intensive. In the first case, it is necessary to adapt the rules to the new language. In the second case, a new training corpus has to be tagged manually for retraining.

In this paper, we compare a new unsupervised approach to sentence boundary detection by Kiss & Strunk [2] with the results of a previous evaluation of three

different systems on English and Portuguese documents [3] carried out by Silla Jr. & Kaestner. The three previous systems are described in the next section.

2 Description of the Systems

2.1 RE System

The first system tested was the RE (Regular Expressions) system[3] developed by Silla Jr. & Kaestner for English; cf. [3]. It was chosen as a representative of the fixed rules approach. The system considers the context where each possible end-of-sentence marker occurs within the document. It uses a database of regular expressions which denote strings that contain punctuation marks but don't indicate the end of a sentence, like abbreviations, e-mail addresses, URLs, etc.

In order to identify sentence boundaries, the system scans the text until it finds a period (.). It then analyzes the preceding string; if this string matches some regular expression, the system concludes that the period is not an end-of-sentence marker and advances to the next period. If the preceding string doesn't match any regular expression, the system considers the string after the period. If it doesn't find any matching regular expression for this string, either, it concludes that the period indicates a sentence boundary. The procedure is repeated until the entire document has been analyzed. The system is also able to deal with ellipses (. . .).

In order to adapt the system to Brazilian Portuguese, 240 new regular expressions containing abbreviations for the new language had to be added.

2.2 MxTerminator

The MxTerminator system[4] was developed by Reynar and Ratnaparkhi [4] at the University of Pennsylvania. It uses a supervised machine learning approach called maximum entropy modelling. From a corpus in which the sentences have been identified manually, the model learns to decide for each instance of period (.), exclamation mark (!) and question mark (?) whether it marks the end of a sentence or not.

The training process is robust and doesn't require any additional linguistic information. During training, the system learns probabilistic contextual features from the training corpus that can be used to identify sentence boundaries with high accuracy, such as e.g. the prefix and suffix occurring around a potential sentence boundary symbol, the preceding and following word, capitalization information, etc. It also induces a list of abbreviations from the training corpus by considering as an abbreviation every token in the training set that contains a possible end-of-sentence symbol but does not indicate a sentence boundary.

The system then uses the contextual features and the abbreviation list learned during training to calculate the probability that a possible end-of-sentence marker in a test corpus indeed indicates a sentence boundary or not.

[3] Available from: http://www.ppgia.pucpr.br/∼silla/softwares/yasd.zip.
[4] Available from: ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz.

The procedure to adapt MxTerminator to Brazilian Portuguese was quite simple because the system only requires a text file of any size that must contain one sentence per line as training corpus.

2.3 Satz

The Satz system[5] was developed by Palmer and Hearst [5] at the University of California in Berkeley. It is a supervised approach that uses estimates of the part-of-speech distribution of the words surrounding potential end-of-sentence punctuation marks as input to a machine learning algorithm. The part-of-speech information is derived from a lexicon that contains part-of-speech frequency data. In case a word is not in the lexicon, a part-of-speech distribution is estimated by different guessing heuristics. In addition, Satz also uses an abbreviation list and capitalization information. After training the system on a small training and a small cross-validation corpus, it can then be used on new documents to detect sentence boundaries. The system can work with any kind of machine learning approach in principle. Palmer & Hearst's original results [5] were obtained using neural networks and the C4.5 decision tree classifier.

For our own evaluation reported in section 4 we employed a re-implementation of the Satz system in Java by Silla Jr. & Kaestner, which uses J4.8 – a Java version of the C4.5 decision tree induction algorithm. The system had to be re-implemented because of problems with accented characters in Portuguese which had occurred with the original version. However, this re-implementation alone was not enough to adapt the system. Silla Jr. & Kaestner also had to create a small training corpus and a new lexicon with part-of-speech information.

3 The Unsupervised System by Kiss & Strunk

The unsupervised system by Kiss & Strunk (subsequently abbreviated as KS)[6] combines type-based and token-based classification[7] in a two-stage approach. It only has to be supplied with the test corpus and does not need further training data, a lexicon, or a list of abbreviations. Instead it uses the test corpus itself as a training corpus on the fly. The system is multilingual in the sense that it is supposed to work for all languages with an alphabetic script in which the period is used to mark both abbreviations and sentence boundaries.

Sentence boundary disambiguation lends itself to a two-stage approach combining type-based and token-based classifiers because in many languages the token-final period (.), the most frequently used sentence boundary marker, is ambiguous in the following way: It can either indicate an abbreviation, a sentence boundary, or an abbreviation at the end of a sentence in which case the

[5] Available from: http://elib.cs.berkeley.edu/src/satz/

[6] The KS system is based on an earlier system described in [6] and [7].

[7] We define a classifier as type-based if it uses global evidence, e.g. the distribution of a type in a corpus, to classify a type as a whole. In contrast, a token-based classifier determines a class for each individual token based on its local context.

period performs a double duty as abbreviation and sentence boundary marker at the same time; cf. [8]. Similar facts hold for ellipses (. . .), a combination of three or more periods that are used to indicate an omission or an omission followed by a sentence boundary in which case the sentence boundary period is also normally haplologically omitted. Abbreviations can be detected very well with a type-based classifier because abbreviations are a (productive) class of lexical items, i.e. all instances of abbreviation types such as *e.g.* or *etc.* are abbreviations regardless of what context they occur in as individual tokens. Moreover, any periods that follow instances of types that have been identified as non-abbreviations by the type-based classifier can safely be classified as sentence boundary markers: If we know that a token with a final period is an ordinary word and not an abbreviation, it is clear that the period following it is a sentence boundary marker. The first stage of the KS system therefore consists of a type-based classifier that separates all word types in the test corpus into the three classes: abbreviation, ellipsis, and ordinary word. Most sentence boundaries are already detected by this type-based first stage. It is described in section 3.1.

The token-based second stage of the KS system improves on the initial classification of the periods in the test corpus performed by the type-based first stage. It re-examines the initial annotation and reclassifies certain cases that can only be decided by token-based classification in principle or present difficulties for the type-based classifier. Whether an abbreviations or an ellipsis is followed by a sentence boundary cannot be decided by a type-based algorithm at all because instances of one and the same abbreviation type – such as the English *etc.* – can be followed by a sentence boundary in one case and occur in the middle of a sentence in another case. The token-based stage therefore decides for all abbreviation and ellipsis tokens in the test corpus whether they precede a sentence boundary or not. In addition, the token-based stage is also used to correct the initial classification for certain subclasses of abbreviations, namely initials – such as in *J. Bond* – and ordinal numbers – such as in the German example *3. März* ("third of March"), which are less amenable to a type-based approach because of problems with homography. The token-based second stage of the KS system is described in section 3.2.

3.1 Initial Type-Based Classification

The type-based classification of the KS system is based on the task of abbreviation detection. By finding all abbreviation types in a test corpus, the system is also able to detect a large portion of the sentence boundaries in the corpus by classifying all periods following non-abbreviation types as sentence boundary markers. Kiss & Strunk assume that abbreviation detection is a manageable subproblem of sentence boundary detection and may also be useful in itself in that dynamically generated lists of abbreviations could be used in subsequent natural language processing tasks.

In their approach, Kiss & Strunk concentrate on the following three characteristics of typical abbreviations:

1. *Strong collocational dependence:* Abbreviations always occur with a final period.[8]

2. *Brevity:* Abbreviations tend to be short.

3. *Internal periods:* Many abbreviations contain additional internal periods.

As these three characteristics do not change for each individual instance of a type, they can be combined in a type-based approach to abbreviation detection.

The criterion of *strong collocational dependence* expresses the intuition that an abbreviation and the final period marking it as such form a tight unit in that an ordinary abbreviation should never occur without a following period. Kiss & Strunk implement this intuition using a modification of Dunning's log-likelihood ratio for collocation detection described in [9]. They use a log-likelihood ratio to compare the probabilities of the following two hypotheses: The null hypothesis H_0 shown in (1) assumes that a type w is not an abbreviation and that therefore the probability of a period occurring after this type is equal to the unconditional probability of occurrence of the period.[9]

$$\text{Null hypothesis } H_0: \qquad P(\bullet|w) = P_{MLE}(\bullet) = \frac{count(\bullet)}{N} \tag{1}$$

The alternative hypothesis assumes that the type w in question is indeed an abbreviation and therefore (almost) always occurs with a following period. The conditional probability of a period given w is therefore taken to be 0.99, i.e. almost one, cf. equation (2).

$$\text{Alternative hypothesis } H_A: \qquad P(\bullet|w) = 0.99 \tag{2}$$

The KS system uses the actual number of occurrences of each type in the test corpus with and without a following period to calculate the probabilities for the two hypotheses with the binomial distribution. The two probabilities are compared using the formula in (3).

$$\log \lambda = -2 \log \frac{P_{binom}(H_0)}{P_{binom}(H_A)} \tag{3}$$

The list of candidate types is sorted according to the calculated log-likelihood values. A type with a higher $\log \lambda$ value is more likely to be an abbreviation according to the criterion of *strong collocational dependence* than all types with lower values. The left half of Table 1 shows a section of this sorted list from

[8] If abbreviations do not have to occur with a final period in a certain language or certain types of abbreviations do not have to, the problem of deciding between the end-of-sentence marker and the abbreviation marker does not occur in this language or for these types of abbreviations.

[9] *MLE* stands for maximum likelihood estimation. N is the number of tokens in the test corpus.

an English test corpus. Some true abbreviations in this table are either ranked lower than non-abbreviations (written in italics) or receive the same log λ values as non-abbreviations. The criterion of *strong collocational dependence* alone is thus not sufficient to separate abbreviations from non-abbreviations.

Table 1. Candidate list from an English test corpus

Candidate type	count(w, •)	count(w, ¬•)	Original log λ	Final sorting	Final log λ
n.h	5	0	28.08	n.h	7.60
u.s.a	5	0	28.08	a.g	6.08
alex	8	2	26.75	m.j	4.56
ounces	4	0	22.46	u.n	4.56
a.g	4	0	22.46	u.s.a	4.19
ga	4	0	22.46	ga	3.04
vt	4	0	22.46	vt	3.04
ore	5	1	18.99	ore	0.32
1990s	5	1	18.99	reps	0.31
mo	8	3	17.67	mo	0.30
m.j	3	0	16.85	*1990s*	0.26
depositor	3	0	16.85	*ounces*	0.06
reps	3	0	16.85	alex	0.03
u.n	3	0	16.85	*depositor*	0.00

The calculated log-likelihood values are therefore taken as a starting point and multiplied with additional factors to obtain an improved sorting of the candidate types. Table 1 confirms that abbreviations tend to be rather short. The factor F_{length} in (4) expresses this intuition and gives an exponentially growing penalty to longer candidate types.

$$F_{length} = \frac{1}{e^{length(w)}} \tag{4}$$

Kiss & Strunk define $length(w)$ as the length of candidate type w minus the number of internal periods in w because internal periods are actually good evidence in favor of a classification as abbreviation and should not lead to a higher penalty by the length factor. Instead, the KS system rewards internal periods with the factor given in (5).

$$F_{period} = \text{number of internal periods} + 1 \tag{5}$$

The scaled log-likelihood ratio proposed by Kiss & Strunk has the advantage that it makes abbreviation detection more robust. The algorithm does not exclude a candidate from being classified as an abbreviation just because it has occurred without a final period once or twice in the whole corpus when there is otherwise good evidence that it is a true abbreviation. For most languages, this increased robustness is unproblematic because almost all ordinary words occur without a period a sufficient number of times. However, for some languages

the log-likelihood ratio in (3) is not restrictive enough. One example are verb-final languages – such as Turkish – where certain very common verbs happen to appear at the end of a sentence most of the time. In such a case, the scaled log-likelihood ratio described so far runs into difficulties because it mistakes the occurrences of these verbs without a period as exceptions. To remedy this problem, the calculated log λ values are additionally multiplied by a third factor that penalizes occurrences without a final period exponentially, cf. equation (6).

$$F_{penalty} = \frac{1}{length(w)^{count(w,\neg\bullet)}} \tag{6}$$

In order to perform the classification into abbreviations and non-abbreviations, the calculated log λ values for all candidate types are multiplied with all three factors. The resulting final values are then compared with a threshold value. All candidates that attain a value greater or equal to the threshold value are classified as abbreviation types all others as non-abbreviation types, cf. (7).

$$\begin{array}{l}\text{For each } w\text{:}\\ \text{If } \log\lambda(w)\times F_{length}\times F_{periods}\times F_{penalty}\geq 0.3 \rightarrow w \text{ is an abbreviation.}\\ \text{If } \log\lambda(w)\times F_{length}\times F_{periods}\times F_{penalty}< 0.3 \rightarrow w \text{ is not an abbreviation.}\end{array} \tag{7}$$

The threshold value 0.3 has been determined experimentally by looking at the sorted list of candidates extracted from a development corpus which was built from a 10 MB part of the Wall Street Journal corpus of American English. Kiss & Strunk assume that the threshold value will not vary much for different languages and corpora and the value 0.3 can thus be used on new corpora and languages without the need for additional manual experiments.[10] The scaling factors have also been derived in experiments measuring their effect on the goodness of the sorting of the candidate list.

The last two columns in Table 1 show the final scaled log λ values of the candidates and the resulting sorting. Multiplication with the three factors has led to a cleaner separation of the candidates into abbreviations and non-abbreviations.

3.2 Token-Based Reclassification

In the token-based reclassification stage of the KS system, all tokens with a final period are re-examined and possibly reclassified. The evidence for this reclassification comes from the immediate right context of the period that is re-examined.[11]

The token-based stage treats different classes of candidates such as abbreviations, ellipses, initials, and ordinal numbers in different ways. However, the reclassification of all the different classes involves the same kinds of evidence combined in slightly different ways.

[10] This view is confirmed by a more detailed evaluation in [2].

[11] If the next token following the period is separated from it by empty lines, up to three new line tokens are ignored, i.e. *etc.* \n \n *This* is treated as *etc. This.*

One type of evidence that is usually taken to be very fundamental for sentence boundary detection, namely capitalization, is only used as secondary evidence during reclassification in the KS system. Moreover, the *orthographic decision heuristic* used is quite cautious, which makes the system very robust against capitalization errors and enables it to process single-case corpora with almost the same accuracy as mixed-case corpora; cf. [2]. As data for the *orthographic decision heuristic*, the capitalization behavior of all types in the test corpus is recorded. For each type, it is counted how often it occurs with an uppercase first letter and how often with a lowercase first letter. Moreover, it is also determined on the basis of the initial annotation from the first stage how often every type occurs upper- and lowercased after a sure sentence boundary[12] and within a sentence. The following is the pseudo-code for the *orthographic decision heuristic*:

```
function DECIDE_ORTHOGRAPHIC (TOKEN):
   if TOKEN has uppercase first letter:
       if TOKEN ever occurs with lowercase first letter:
           if TOKEN never occurs with uppercase first letter
           sentence internally:
               Return sentence_boundary
           else
               Return undecided
       else
           Return undecided

   else if TOKEN has lowercase first letter:
       if (TOKEN ever occurs with uppercase first letter)
       or (never occurs with lowercase first letter after
       a sentence boundary):
           Return no_sentence_boundary
       else
           Return undecided
```

The *orthographic decision heuristic* is especially cautious in two cases: First, if a type also occurs with an uppercase first letter within a sentence, as is usually the case with proper names, it is no longer counted as evidence for a preceding sentence boundary if an instance of this type follows a period. Second, if a type also occurs in lower case after a sure sentence boundary, it might be a mathematical symbol or a special word such as *amnesty international* that is always written with a lowercase first letter. This type is then no longer counted as evidence against a sentence boundary if it follows a period.[13]

The second type of evidence that the system relies on during the token-based stage is collocational data. It is often assumed that there are no strong local dependencies between the end of one and the beginning of the following sentence; cf. e.g. page 195 in [10]. If there is a strong collocational dependence between two

[12] This means all periods following a type classified as an ordinary word that is longer than one letter, i.e. no possible initial, and is not a number written in digits.

[13] This also enables the KS system to classify all-lowercase corpora without bad reclassification by the *orthographic decision heuristic*.

types – such as e.g. between an initial and a following last name – this is good evidence against an intervening sentence boundary. The KS system therefore employs the standard log-likelihood ratio for collocation detection described in [9] to calculate the dependence between two types.[14] If the log-likelihood ratio yields a value greater or equal to 7.88, the two types are considered as collocates and as evidence against an intervening sentence boundary.[15]

However, collocational data is also used as evidence in favor of a sentence boundary. For this purpose, the collocational dependence between every type in the test corpus and the abstract type *preceding sentence boundary* is calculated in order to generate a list of frequent sentence starters on the fly. The counts used in these calculations are based on all clear sentence boundaries detected by the type-based first stage. All types for which Dunning's log-likelihood ratio yields a value of at least 30 are considered as frequent sentence starters and regarded as evidence for a sentence boundary if they occur after a period and are written with an uppercase first letter.[16]

The main question for all tokens classified as abbreviations by the type-based first stage and all ellipses is whether they precede a sentence boundary. A sentence boundary after these two classes of candidate tokens is assumed by the KS system if the *orthographic decision heuristic* decides in favor of a sentence boundary or the token following the period is a capitalized frequent sentence starter. However, only abbreviations that are longer than one letter and thus not possibly initials are reclassified in this way. Initials present special problems and are therefore reclassified differently.

Initials are a subclass of abbreviations consisting of a *single* letter followed by a period. As there are only about thirty different letters in the average Latin-derived alphabet, the likelihood of being a homograph of a non-abbreviation is very high for initials, consider e.g. the Portuguese articles *o* and *a* or the Swedish preposition *i*. Initials are therefore often not detected by the type-based first stage of the KS system. For this reason, every single letter followed by a token-final period is treated as a possible initial during the token-based reclassification – regardless of whether it has been classified as an abbreviation or not by the type-based stage. Luckily, initials are very often part of a complex name and can be identified using collocational evidence. If a possible initial forms a collocation with the following token and the following token is not a frequent sentence starter, the period in between is reclassified as an abbreviation marker. Alternatively, if the *orthographic decision heuristic* decides against a sentence boundary on the basis of the token following the possible initial, the period is also reclassified as an abbreviation period. Last but not least, if the *orthographic decision heuristic* returns *undecided* and the type following the possible initial always occurs with an uppercase first letter, it is assumed to be a name and the period between the two tokens is again classified as an abbreviation marker.

[14] All numbers written in digits are folded into one abstract type *##number##*.

[15] This value was chosen because it represents a confidence degree of 99.95 % according to the χ^2 distribution and worked well on our English development corpus.

[16] The threshold value 30 was determined experimentally on our development corpus.

In many languages such as e.g. German, ordinal numbers written in digits are also marked by a token-final period. However, as every numeric type can also be used as a cardinal number, it cannot be decided by a type-based algorithm whether a period after a number is an abbreviation period or a sentence boundary marker. Numbers are therefore treated in the same way as initials. If the token following a number with a final period forms a collocation with the abstract type *##number##* and is not a frequent sentence starter, the period in between is classified as an abbreviation marker. The same conclusion is reached if the *orthographic decision heuristic* decides against a sentence boundary. In other languages such as English and Portuguese on which we did the evaluation for this paper, ordinal numbers are usually not marked with a period. For these languages, the detection of ordinal numbers can be turned off. The results of the test runs of the KS system reported in section 4 were determined with the detection of ordinal numbers switched off. As the detection of ordinal numbers is a major feature of the KS system we have described it here nonetheless.

4 Experiments and Results

The RE system, MxTerminator, and Satz had already been evaluated by Silla Jr. & Kaestner in a previous comparative study [3] on English and Portuguese documents. For the current paper, we have used the same two test corpora in English and Portuguese to evaluate the unsupervised system by Kiss & Strunk. This allows for a direct comparison of the performance of all four systems.[17]

In order to perform the experiments, each of the test documents had its sentence boundaries tagged manually. The different systems were then run on these test documents and the resulting annotation was compared to the reference annotation. We use the following performance measures: *Precision* is calculated as the percentage of correctly classified sentence boundaries, i.e. the number of sentence boundaries *correctly* identified divided by the number of sentence boundaries identified. *Recall* indicates the percentage of sentence boundaries present in the document that were actually found by a particular system, i.e. the number of sentence boundaries correctly identified divided by the number of sentence boundaries present in the reference annotation. The *f-measure* combines precision and recall in a single metric: the harmonic mean. As an indication of the difficulty of the sentence boundary detection task on the two test corpora, we compare the results of the four systems with a simple baseline, which assumes that every token-final period indicates a sentence boundary.

As the test corpus for English, we used part of the TIPSTER document collection from the Text Retrieval Conference, which contains articles from the Wall Street Journal (TREC reference number: WSJ-910130). This corpus comprises 156 documents of different sizes, totaling 3,554 sentences. We performed two different test runs with the unsupervised system by Kiss & Strunk: For the first one, we used the individual files containing the WSJ articles; for the second

[17] Kiss & Strunk have carried out a more extensive evaluation of their system on further languages and genres which is reported in [2].

one, we provided the system with all articles pasted together in a single file. This was necessary to ensure a fair comparison between the different systems because the KS system does not use any additional training data but instead learns from the test corpus on the fly. When the test corpora used are very small, the KS system is likely to suffer from data sparseness.[18]

Table 2 shows the results achieved on the English test corpus by the four systems. When the KS system is run on the corpus as a single file, it produces only slightly worse results than Satz – a supervised system which uses a language specific lexicon and abbreviation list – and the RE system – which has been specifically tailored to English newspaper texts – and it is even slightly better than MxTerminator – a more straightforward machine learning system. When the KS system is tested on the individual WSJ articles, which sometimes contain less than ten sentences, performance drops considerably due to data-sparseness but is still much better than the baseline.

Table 2. Results on the TIPSTER document collection (English)

System	Precision	Recall	F-Measure
Baseline	30,29 %	50,61 %	37,89 %
KS (individual files)	80,43 %	83,40 %	81,88 %
MxTerminator	91,19 %	91,25 %	91,22 %
KS (single file)	90,70 %	92,34 %	91,51 %
RE	92,39 %	91,18 %	91,78 %
Satz	98,67 %	85,98 %	91,88 %

For Portuguese, we used the Lacio-Web Corpus [11], which contains 21,822 sentences in all. The systems were tested on this corpus using 10-fold cross-validation. The results achieved are presented in Table 3.

Table 3. Results on the Lacio-Web document collection (Portuguese)

System	Precision	Recall	F-Measure
Baseline	85,40 %	92,25 %	88,69 %
RE	91,80 %	88,02 %	89,87 %
MxTerminator	96,31 %	96,63 %	96,46 %
KS	97,58 %	96,87 %	97,22 %
Satz	99,59 %	98,74 %	99,16 %

The results of the KS system on Portuguese are better than those of MxTerminator and the RE system. The KS system is only second to Satz. It has to be kept in mind, however, that Satz, MxTerminator, and the RE system had

[18] This could be remedied by equipping the KS system with a kind of memory function, so that it is able to remember data from previous test runs.

to be customized before applying them to the Portuguese corpus, while the KS system was used as is both for English and Portuguese.

5 Conclusions

We have described an unsupervised approach to sentence boundary detection developed by Kiss & Strunk and have presented the results of a comparative evaluation of this approach and three earlier systems – the RE system, MxTerminator, and Satz – on English and Portuguese corpora. We conclude that the unsupervised approach can be very useful since it can be used out of the box for new languages and genres, while the supervised or rule-based approaches have to be adapted by hand or need retraining, which requires resources that are not always available. Moreover, the performance of the unsupervised KS system is only slightly worse and sometimes even better than that of the other systems.

Acknowledgments

We would like to thank Adwait Ratnaparkhi for sending us the MxTerminator system, and Marti A. Hearst for providing the original files used by Satz.

References

1. Lyman, P., Varian, H.R.: How much information. Retrieved from http://www.sims.berkeley.edu/how-much-info-2003 on [01/19/2004] (2003)
2. Kiss, T., Strunk, J.: Multilingual unsupervised sentence boundary detection. http://www.linguistics.rub.de/∼strunk/ks2005FINAL.pdf (Under Review)
3. Silla Jr., C.N., Kaestner, C.A.A.: An analysis of sentence boundary detection systems for English and Portuguese documents. In Gelbukh, A., ed.: Computational Linguistics and Intelligent Text Processing. Volume 2945 of Lecture Notes in Computer Science, CAU, Seoul, Korea, Springer Verlag (2004) 135–141
4. Reynar, J., Ratnaparkhi, A.: A maximum entropy approach to identifying sentence boundaries. In: Proceedings of the Fifth Conference on Applied Natural Language Processing. (1997) 16–19
5. Palmer, D.D., Hearst, M.A.: Adaptive multilingual sentence boundary disambiguation. Computational Linguistics **23/2** (1997) 241–267
6. Kiss, T., Strunk, J.: Scaled log likelihood ratios for the detection of abbreviations in text corpora, *Proceedings of COLING 2002, Taipei* (2002) 1228–1232
7. Kiss, T., Strunk, J.: Viewing sentence boundary detection as collocation identification, *Proceedings of KONVENS 2002, Saarbrücken* (2002) 75–82
8. Nunberg, G.: The Linguistics of Punctuation. CSLI Lecture Notes Number 18. Center for the Study of Language and Information, Stanford, California (1990)
9. Dunning, T.: Accurate methods for the statistics of surprise and coincidence. Computational Linguistics **19/1** (1993) 61–74
10. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge/London (1999)
11. Aluisio, S.M., Pinheiro, G.M., Finger, M., Nunes, M.G.V., Tagnin, S.E.: The Lacio-Web Project: Overview and issues in Brazilian Portuguese corpora creation. In: Proceedings of Corpus Linguistics 2003. (2003) 14–21