Unsupervised Multilingual Sentence Boundary Detection

Tibor Kiss[*]
Ruhr-Universität Bochum

Jan Strunk[†]
Ruhr-Universität Bochum

In this article, we present a language-independent, unsupervised approach to sentence boundary detection. It is based on the assumption that a large number of ambiguities in the determination of sentence boundaries can be eliminated once abbreviations have been identified. Instead of relying on orthographic clues, as is usually done, the proposed system is able to detect abbreviations with high accuracy using three criteria that only require information about the candidate type itself and are independent of context: Abbreviations can be defined as a very tight collocation consisting of a truncated word and a final period, abbreviations are usually short, and abbreviations sometimes contain internal periods. We also show the potential of collocational evidence for two other important subtasks of sentence boundary disambiguation, namely the detection of initials and ordinal numbers. The proposed system has been tested extensively on eleven different languages and on different text genres. It achieves good results without any further amendments or language-specific resources. We evaluate its performance against three different baselines and compare it to other systems for sentence boundary detection proposed in the literature.

1 Introduction

The sentence is a fundamental and relatively well-understood unit in theoretical and computational linguistics. Syntactic processing is generally performed on a sentence-by-sentence basis, and many processes are constrained by this abstract concept in that they may be active inside a sentence but not between consecutive sentences. Among these, we find collocations, idioms, anaphora, as well as variable binding and quantification. Given that these processes are crucially constrained by sentence boundaries, the successful determination of these boundaries is a prerequisite for proper sentence processing. Sentence boundary detection is not a trivial task, though. Graphemes often serve more than one purpose in writing systems. The period, which is employed as sentence boundary marker, is no exception to this rule. It is also used to mark abbreviations, initials, ordinal numbers, and ellipses. Moreover, a period can be used to mark an abbreviation and a sentence boundary at the same time. In such cases, the second period is haplologically omitted and only one period is used as end-of-sentence and abbreviation marker.[1] Sentence boundary detection thus has to be considered as an instance of ambiguity resolution. The ambiguity of the period is illustrated by example (1).

* Sprachwissenschaftliches Institut, Ruhr-Universität Bochum, 44780 Bochum, Germany, E-mail: tibor@linguistics.rub.de
† Sprachwissenschaftliches Institut, Ruhr-Universität Bochum, 44780 Bochum, Germany, E-mail: strunk@linguistics.rub.de
1 See Nunberg (1990) for a linguistic discussion of punctuation and the ambiguity of the period.

Example 1
CELLULAR COMMUNICATIONS INC. sold 1,550,000 common shares at $21.75 each yesterday, according to lead underwriter L.F. Rothschild & Co. (cited from WSJ 05/29/1987)

Periods that form part of an abbreviation but are taken to be end-of-sentence markers or vice versa do not only introduce errors in the determination of sentence boundaries. Segmentation errors propagate into further components which rely on accurate sentence segmentation and subsequent analyses are most likely affected negatively. Walker et al. (2001), for example, stress the importance of correct sentence boundary disambiguation for machine translation and Kiss and Strunk (2002b) show that errors in sentence boundary detection lead to a higher error rate in part-of-speech tagging.

In this paper, we present an approach to sentence boundary detection that builds on language-independent methods and determines sentence boundaries with high accuracy. It does not make use of additional annotations, part-of-speech tagging, or precompiled lists to support sentence boundary detection but extracts all necessary data from the corpus to be segmented. Also, it does not use orthographic information as primary evidence and is thus suited to process single-case text. It focuses on robustness and flexibility in that it can be applied with good results to a variety of languages without any further adjustments. At the same time, the modular structure of the proposed system makes it possible in principle to integrate language-specific methods and clues to further improve its accuracy. The basic algorithm has been determined experimentally on the basis of an unannotated development corpus of English. We have applied the resulting system to further corpora of English text as well as to corpora from ten other languages: Brazilian Portuguese, Dutch, Estonian, French, German, Italian, Norwegian, Spanish, Swedish, and Turkish. Without further additions or amendments to the system produced through experimentation on the development corpus, the mean accuracy of sentence boundary detection on newspaper corpora in eleven languages is 98.74 %.

Sentence boundary detection has received some attention in the past decade, as can be witnessed by approaches such as Grefenstette and Tapanainen (1994), Palmer and Hearst (1997), Reynar and Ratnaparkhi (1997), and Mikheev (2002), amongst others. It is a common property of these approaches that they have been applied to a small range of languages only, usually covering one to three languages. Although these methods are very successful when applied to these individual languages, one of which is usually English, concentrating on a small set of languages is problematic in two respects. First, it remains unclear how well the suggested methods operate if they are tested on a wider sample of languages. Second, the complexity of the task itself may be underestimated if only familiar languages are considered. Mikheev (2002, page 314), for example, proposes that an ideal upper bound error rate for sentence boundary detection could be 0.02 % because he estimates disagreement of human annotators on this task at about 1 in 5000 cases. But Mikheev's assessment seems to presume that language-specific systems for sentence boundary detection should be compared to the proficiency of individual speakers who are fluent in the languages under evaluation. Surely, a Russian speaker with no knowledge of German would reach an accuracy of 99.98 % only by chance, if at all, on the task of segmenting a German text into sentences. If the present system is compared to the performance of human speakers, it should be compared to an idealized speaker who is able to detect sentence boundaries accurately even for languages that he/she has no knowledge of. Under these circumstances, the estimated lower bound for sentence boundaries detection will presumably be higher than Mikheev's estimate.

We approach sentence boundary detection by first determining possible abbreviations in the text. Quantitatively, abbreviations are a major source of ambiguities in sentence boundary detection since they often constitute up to 30 % of the possible candidates for sentence boundaries in running text; see section 6.1. While the concept 'sentence boundary' remains elusive in that typical, cross-linguistically valid, as well as robust properties of a sentence boundary cannot easily be defined, the same does not hold for abbreviations. The end of a sentence cannot easily be characterized as either appearing after a particular word, between two particular words, after a particular word class, or in between two particular word classes. But, as we will show, an abbreviation can be cross-linguistically characterized in such a way. This is so because the end-of-sentence marker cannot be linked to an intrinsic property of the sentence while a period marking an abbreviation can be related to the abbreviated word itself.

It is our basic assumption that abbreviations are collocations of the truncated word and the following period, and hence, that methods for the detection of collocations can be successfully applied to abbreviation detection. Firth (1957, page 181) characterizes the collocations of a word as "statements of the habitual or customary places of that word." In languages that mark abbreviations with a following period, one could say that the abbreviation is habitually made up of a truncated word (or sequence of words) and a following period. But this might even be too weak a formulation. While typical elements of a collocation can also appear together with other words, the abbreviation is strongly tied to the following period. Ideally, in the absence of homography and typing errors, an abbreviation should *always* end in a final period. Hence, we characterize an abbreviation as a very strict collocation and use standard techniques for the detection of collocations. These techniques will be modified appropriately to account for the stricter tie between an abbreviated word and the following period. It should be clear from the outset that abbreviations cannot simply be handled by listing them because they form a productive and hence open word class; see also Müller, Amerl, and Natalis (1980, pages 52f.) and Mikheev (2002, page 291). We corroborate this fact with an experiment in section 6.4.4.

We offer a formal characterization of abbreviations in terms of three major properties, which only rely on the candidate word type itself and not on the local context in which an instance of the candidate type appears. First, as was already mentioned, an abbreviation looks like a very tight collocation in that the abbreviated word preceding the period and the period itself form a close bond. Second, abbreviations have the tendency to be rather short. This does not mean that we have to assume a fixed upper bound for the length of a possible abbreviation, but that the likelihood of being an abbreviation declines if candidates become longer. Using the length of a candidate as a counterbalance to the collocational bond between candidate and final period allows our method to identify quite long abbreviations, as long as the collocational bond between the candidate type and the period is very strong. As a third characteristic property, we have identified the occurrence of word-internal periods contained in many abbreviations. While we have determined the aforementioned properties experimentally, we believe that they indeed represent crucial traits of abbreviations.

Using just these three characteristics, our system is able to detect abbreviations with a very high mean accuracy of 99.38 % on newspaper corpora in eleven languages. The effectiveness of the three properties is further corroborated by an experiment we have carried out with a log-linear classifier; compare section 6.4.6. The reported figure does not include initials and ordinal numbers because these subclasses of abbreviations cannot be discovered using these characteristics and have to be treated differently. The *complete* system with special heuristics for initials and ordinal numbers achieves an accuracy of 99.20 % for the detection of abbreviations, initials, and ordinal numbers.

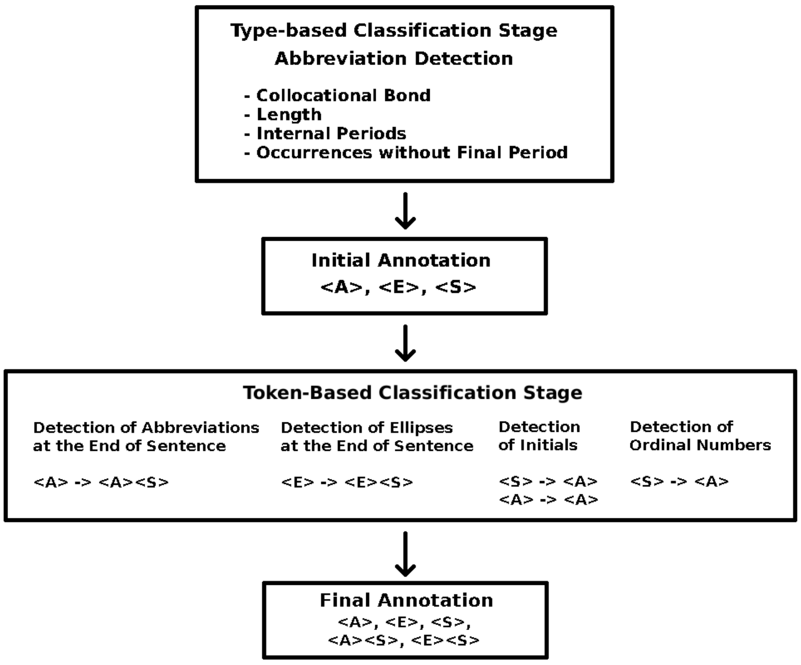

Figure 1
Architecture of the Punkt System

The determination of abbreviation types already yields a large percentage of all sentence boundaries because all periods occurring after non-abbreviation types can be classified as end-of-sentence markers. Such a disambiguation on the type level, however, is insufficient by itself because it still has to be determined for every period following an abbreviation whether it serves as a sentence boundary marker at the same time. This observation suggests a treatment of sentence boundary detection which is both type and token-based. We define a classifier as type-based if it uses global evidence, for example, 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. The detection of initials and of ordinal numbers, which are represented by digits followed by a period in several languages, also requires the application of token-based methods because these subclasses of abbreviations are problematic for type-based methods.

The detection of sentence boundaries and abbreviations thus lends itself to a two-stage approach combining type-based and token-based classifiers. In the first stage, a resolution is performed on the type level to detect abbreviation types and ordinary word types. After this stage, the corpus receives an intermediate annotation where all instances of abbreviations detected by the first stage are marked as such with the tag <A> and all ellipses with the tag <E>. All periods following non-abbreviations are assumed to be sentence boundary markers and receive the annotation <S>. The second, token-based stage employs additional heuristics on the basis of the intermediate annotation to refine and correct the output of the first classifier for each individual token. The token-based classifier is particularly suited to determine abbreviations and ellipses at the end of sentence giving them the final annotation <A><S> or <E><S>. But it is also used to correct the intermediate annotation by detecting initials and ordinal numbers which cannot easily be recognized with type-based methods and thus often receive the wrong annotation from the first stage. The overall architecture of the present system, which we have baptized **Punkt** (German for period), is given in Figure 1.

The present article is structured as follows: Likelihood ratios can be considered the heart of the present proposal. Both the type-based and the token-based classifiers make use of likelihood ratios to determine collocational bonds between a possible abbreviation and its final period, between the sentence boundary period and a word following it, and between words which surround a period. Section 2 introduces the concept of a likelihood ratio and discusses the specific properties of the likelihood ratios employed by Punkt. Section 3 describes the type-based classification stage, while section 4 introduces the token-based re-classification methods. Section 5 gives an account of how Punkt was developed and how we determined some necessary parameters. The experiments carried out with the present system are discussed in section 6. In section 7, we compare Punkt to other sentence boundary detection systems proposed in the literature.

2 Likelihood Ratios

Punkt employs likelihood ratios to determine collocational ties in the type-based as well as in the token-based stage. The usefulness of likelihood ratios for collocation detection has been made explicit in Dunning (1993) and has been confirmed by an evaluation of various collocation detection methods carried out in Evert and Krenn (2001). Kiss and Strunk (2002a) and (2002b) characterize abbreviations as collocations and use Dunning's log-likelihood ratio (log λ) to detect them on the type level. The present proposal differs from Kiss and Strunk's earlier suggestion in employing a highly modified log-likelihood ratio for abbreviation detection in the type-based stage. The reasons for this divergence will be discussed in section 2.1. In the token-based stage, we employ Dunning's original log λ, but add an additional constraint to make it one-sided. This version of log λ will be described in section 2.2.

2.1 Likelihood Ratios in the Type-Based Stage

The log-likelihood ratio proposed by Dunning (1993) tests whether the probability of a word is dependent on the occurrence of the preceding word type or not. When applied to abbreviations, the following two hypotheses would be compared in a Dunning-style log-likelihood ratio.

$$\text{Null hypothesis } H_0\text{:} \qquad P(\bullet|w) = p = P(\bullet|\neg w) \tag{1}$$

$$\text{Alternative hypothesis } H_A\text{:} \qquad P(\bullet|w) = p_1 \neq p_2 = P(\bullet|\neg w) \tag{2}$$

The null hypothesis in (1) states that the probability of occurrence of a period is not dependent on the preceding word. The alternative hypothesis in (2) assumes a dependency between the period and the preceding word.[2] The probabilities p, p_1, and p_2 in (1) and (2) are determined by maximum-likelihood estimation. The log-likelihood ratio is then determined by the formula in (3) using the binomial distribution to calculate the probabilities of the hypotheses:

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

2 As will become clear in section 2.2, this could even be a negative dependency in the sense that a preceding word makes the occurrence of a following period less likely.

Since the null hypothesis H_0 in Dunning's formulation employs fewer parameters than the alternative hypothesis H_A, and p, p_1, and p_2 are determined by maximum-likelihood estimation, the resulting ratio log λ is asymptotically χ^2-distributed and can thus be used as a test statistic.

Still, we have decided to employ hypotheses which are different from the ones given in (1) and (2) in form and content. The revised null hypothesis given in (4) is quite close to (1) in expressing that the likelihood of the period occurring after a particular word is independent of the occurrence of this word.[3] However, the revised version of the alternative hypothesis in (5) is radically different from the original formulation by Dunning in (2).

$$\text{Revised null hypothesis } H_0\text{:} \qquad P(\bullet|w) = P_{MLE}(\bullet) = \frac{C(\bullet)}{N} \tag{4}$$

$$\text{Revised alternative hypothesis } H_A\text{:} \qquad P(\bullet|w) = 0.99 \tag{5}$$

The formulation of the alternative hypothesis in (5) reflects that we do not only require that a period occurs together with an abbreviated word more often than expected, but instead that a period almost always occurs together with the truncated word. By choosing the value 0.99 instead of 1, we can provide for a certain probability that an abbreviation type sometimes erroneously occurs without a final period in a corpus. The hypothesis in (5) captures our intuitions about abbreviations better than the original version in (2) because it is no longer sufficient that a certain word type appears more often than average with a following period to yield a high log-likelihood score. Instead, the likelihood of a period appearing after a type should be almost 1 in order for it to get assigned a high log-likelihood score and thus a high likelihood of being classified as an abbreviation.

Please note that the probability in (5) is not determined by maximum-likelihood estimation. Also, the revised null hypothesis H_0 does not contain fewer parameters than the revised alternative hypothesis H_A. This means that after feeding the hypotheses (4) and (5) into (3), the resulting log-likelihood ratio is no longer asymptotically χ^2-distributed. However, this is not a disadvantage since the resulting log-likelihood value expresses only one of three crucial properties of abbreviations. Since this value is counterbalanced by other factors and the resulting log-likelihood score thus scaled in various ways, the χ^2 distribution could not be retained anyway, as will become clear in section 3.

2.2 Likelihood Ratios in the Token-Based Stage

In the token-based classification stage, Dunning's log-likelihood ratio is used in two different heuristics. The collocation heuristic takes a pair of words w_1 and w_2 surrounding a period and tests whether a collocational tie exists between them. A positive answer to this question is used as evidence against an intervening sentence boundary. The collocation heuristic is described in section 4.1.2. The frequent sentence starter heuristic, compare section 4.1.3, makes use of the results of the type-based classifier and searches for word types which form a collocation with a preceding sentence boundary, that is, which occur particularly often after end-of-sentence periods.

Dunning's formulation of the log-likelihood ratio is a two-tailed statistical test. For a pair of word types w_1 and w_2, the null hypothesis H_0: $P(w_2 \mid w_1) = p = P(w_2 \mid \neg w_1)$, and the alternative hypothesis H_A: $P(w_2 \mid w_1) = p_1 \neq p_2 = P(w_2 \mid \neg w_1)$, the log λ value

3 N represents the number of tokens in the corpus. C(●) is the number of times a token-final period occurs in the corpus. C(. . .) always signifies the absolute frequency of some element or elements in the corpus.

Table 1
Correct and Incorrect Frequent Sentence Starters

w_1	w_2	$C(w_1)$	$C(w_2)$	$C(w_1, w_2)$	$log\lambda$	
<S>	*ist*	35,775	9,758	182	169.8390	$p_1 << p_2$
<S>	*zu*	35,775	7,643	71	298.8915	$p_1 << p_2$
<S>	*dennoch*	35,775	231	80	221.4709	$p_1 >> p_2$
<S>	*erstens*	35,775	38	21	82.1377	$p_1 >> p_2$

is high if p_1 and p_2 significantly diverge from each other. But in fact, one should only consider those pairs of words as collocations for which p_1 is much higher than p_2. Only the latter case means that w_2 occurs more often than expected after w_1, whereas if p_1 is less than p_2 this means that w_2 occurs less often after w_1 than expected. Manning and Schütze (1999, page 172) comment in their description that "[w]e assume that $p_1 >> p_2$ if Hypothesis 2 [i.e. the alternative hypothesis, TK/JS] is true. The case $p_1 << p_2$ is rare, and we will ignore it here." To us, this step seems to be premature since it ignores that Dunning's log λ is a two-tailed test, where the equation in (6) holds.

$$\log \lambda = 0 \text{ iff } \frac{C(w_2)}{N} = \frac{C(w_1, w_2)}{C(w_1)} \tag{6}$$

If the two sides of the equation in (6) diverge, log λ will take a value greater than 0. In the case which has been considered by Dunning (1993) and discussed in the aforementioned quote from Manning and Schütze (1999), the right hand side of the equation (i.e. $\frac{C(w_1,w_2)}{C(w_1)}$) is larger than the left hand side. A high log λ value thus properly expresses the fact that w_1 and w_2 form a collocation because the occurrence of w_2 after w_1 is more likely than expected from the unconditional likelihood of w_2. If, however, the left side of the equation turns out to be greater than the right side, we still get a log λ value greater than 0, but this time, this indicates that w_2 occurs less often than expected after w_1. This is obviously at odds with the general idea of a collocation.

For collocations in general, the assumption made by Manning and Schütze can indeed be considered safe, since the types w_1 and w_2 are likely to occur only rarely. For this reason, we do not expect a large negative deviation, where $\frac{C(w_1,w_2)}{C(w_1)}$ is significantly smaller than $\frac{C(w_2)}{N}$. However, some of the types that we test for collocational ties in the heuristics of the token-based stage occur very often. The frequent sentence starter heuristic, for example, uses the results of the type-based first stage and tests whether a given word w_2 forms a collocation with a preceding sentence boundary w_1, that is, with the sentence boundary symbol <S> inserted by the type-based classifier. The abstract type 'sentence boundary' (i.e. <S>) may be very frequent in many corpora, as can be witnessed from a sample from a German newspaper corpus, where C(<S>) = 35,775 and N = 847,206. In Table 1, we have tested for four words in the German newspaper corpus whether they are frequent sentence starters or not. The first two words, *ist* (the 3rd-person singular form of the verb *sein* 'to be') and *zu* (infinitive marker or preposition 'to') occur very often, while the latter two words do not occur so often. Neither *ist* nor *zu* should be considered frequent sentence starters, while both *dennoch* ('nevertheless') and *erstens* ('first') are true frequent sentence starters. Yet, all four words receive very high log λ values, since in all cases p_1 diverges significantly from p_2. However, it holds only for *dennoch* and *erstens* that they occur much more often after <S> than elsewhere, while *ist* and *zu* occur much less often than expected after <S>. In order to exclude cases like *ist* and *zu*, for which $p_1 << p_2$ is true, from being detected as collocates of a preceding sentence boundary, we add the constraint in (7) to log λ calculations for the frequent sentence starter heuristic and similarly apply it to the collocation heuristic.

$$< w_1, w_2 > \text{ is a collocation if } \log \lambda > \text{threshold and } \frac{C(w_1,w_2)}{C(w_1)} > \frac{C(w_2)}{N} \quad (7)$$

In general, it seems to be a good idea to add the one-sidedness condition given in (7) to the log-likelihood ratio used for the purpose of collocation detection. The version of the likelihood ratio which is calculated for the type-based classifier in section 3 is not affected by this problem since it does not follow a χ^2 distribution and is in fact no longer two-tailed because of the form of the two hypotheses which are compared.

3 Type-Based Classification

The type-based classification stage of Punkt employs three characteristic properties of abbreviations:

1. Strong collocational dependency: Abbreviations always occur with a final period.[4]
2. Brevity: Abbreviations tend to be short.
3. Internal periods: Quite a few abbreviations contain additional internal periods.

As these three characteristics do not change for each individual instance of a type, we combine them in a type-based approach to abbreviation detection.

The implementation of the first property makes use of a likelihood ratio with the revised hypotheses introduced in (4) and (5). The list of all types which ever occur with a following period in the corpus is sorted according to this likelihood ratio. The resulting value for a type expresses the assumption that this type is more likely to be an abbreviation than all types having lower values.

Table 2
Candidate List from an English Test Corpus

Candidate type	$C(w, \bullet)$	$C(w, \neg\bullet)$	Revised log λ	Final sorting	Scaled 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

4 If all or certain abbreviations do not occur with a final period in a language, the problem of deciding between the sentence boundary and the abbreviation marker does not occur in that language or for these abbreviations.

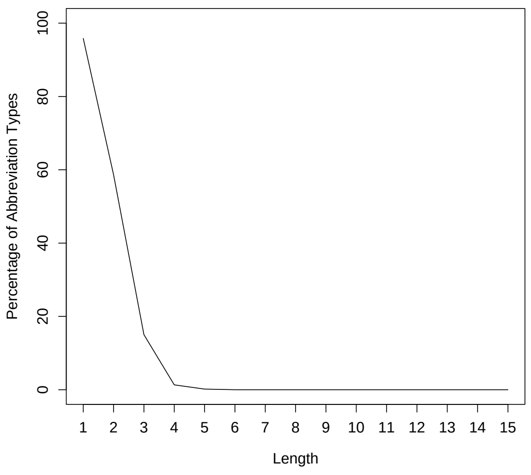

Figure 2
Percentage of All Types of Different Lengths that are Abbreviations in a Dutch Corpus

The left half of Table 2 shows a section of this sorted list, where non-abbreviations are indicated in italics. The figures of occurrence in Table 2, which are drawn from an actual corpus sample, also illustrate a potential problem, namely that most of the candidate types are quite rare. As has been pointed out by Dunning (1993), the calculation of log λ assumes a binomial distribution. It is therefore better suited to deal with sparse data than statistics that are based on a normal distribution, such as for example the t-test. This consideration carries over to the modified log-likelihood ratio employed here.

Some true abbreviations in the left half of Table 2 are either ranked lower than non-abbreviations or receive the same log-likelihood value as non-abbreviations. According to the criterion of strong collocational dependency, *ounces* is a very good candidate for an abbreviation, as it never occurs without a final period in the corpus. The collocational criterion alone is thus not sufficient to detect abbreviations with high precision. Table 2 confirms that abbreviations tend to be rather short: each non-abbreviation is longer than the longest abbreviation. We therefore use brevity as a further characteristic property to counterbalance the likelihood ratio. In contrast to other proposals such as Mikheev (2002, page 299), we refrain from using a fixed maximum length for abbreviations. Instead, we multiply the likelihood ratio for each candidate with an inversely exponential scaling factor derived from the length of that candidate. The length of the candidate is defined as the number of characters in front of the final period minus the number of internal periods, as illustrated in example (8).

$$length(u.s.a.) = 3 \tag{8}$$

We exclude internal periods from the count because they are good evidence that a candidate should be classified as an abbreviation (see below). We thus prevent a counter-intuitive, higher penalty for candidates with internal periods. The exact form of the length factor is given in (9).

$$F_{length} = \frac{1}{e^{length(w)}} \tag{9}$$

We have chosen an inversely exponential scaling factor since it reflects the dependency of the number of abbreviation types on the length of the type. Typically, we find more abbreviation types which are shorter and less abbreviation types which are longer. The validity of this assumption can be witnessed from Figure 2, which shows the dependency between abbreviation length and percentage of abbreviation types in a Dutch newspaper corpus.

In addition to avoiding any higher penalty on the factor F_{length} caused by internal periods, we use another scaling factor $F_{periods}$ as given in (10). This factor expresses the intuition that a higher number of internal periods increases the likelihood that a type is a true abbreviation. The scaling factor has been designed to leave unchanged the values of candidates that do not contain internal periods, while those candidates that contain internal periods receive an extra advantage.

$$F_{periods} = \text{number of internal periods} + 1 \tag{10}$$

Multiplying the log-likelihood ratio with these two scaling factors leads to a significantly better sorting of the candidate list.

The scaled log-likelihood ratio 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 if 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 scaled log-likelihood ratio 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, which penalizes occurrences without a final period exponentially; compare equation (11).

$$F_{penalty} = \frac{1}{length(w)^{C(w,\neg\bullet)}} \tag{11}$$

It should be noted that we use the length of the candidate as a basis for the calculation because the likelihood of existence of homographic non-abbreviations for a given abbreviation type is dependent on the length of the candidate. Homographic non-abbreviations occur particularly often with abbreviations of length 1, and accordingly, there will be no penalty at all. With a length of 2, the penalty factor is still moderate, but increases with length to reflect that longer abbreviations are not very likely to have homographic non-abbreviations. Furthermore, the penalty factor is exponentially scaled. Hence, it will mostly affect candidate types with a high number of occurrences without a final period. As a consequence, a candidate type which occurs six times in all and one time without a final period is much less affected by (11) than a candidate which occurs 600 times in all and 100 times without a final period. This reflects our intuition that homographic non-abbreviations and typing errors can always occur, but that a high number of instances without a final period requires a strong penalty.

As we cannot rely any longer on the asymptotic χ^2 distribution of the log-likelihood ratio (cf. section 2.1), we propose a new threshold value. All candidates above it will be considered abbreviations; all candidates below it will be classified as ordinary words. We use 0.3 as a threshold value – represented by a dashed line in Table 2 – and the classification function defined in (12).

For each candidate word type w:
If $\log \lambda(w) \times F_{length}(w) \times F_{periods}(w) \times F_{penalty}(w) \geq 0.3 \rightarrow w$ is an abbreviation. (12)
If $\log \lambda(w) \times F_{length}(w) \times F_{periods}(w) \times F_{penalty}(w) < 0.3 \rightarrow w$ is not an abbreviation.

The right half of Table 2 shows the final sorting of the candidates after applying the three scaling factors to the log-likelihood value for each candidate. Multiplication with the three factors has led to a much cleaner separation of the true abbreviation types from the non-abbreviations.

We have determined the threshold value experimentally by looking at the sorted list of candidates extracted from a development corpus of American English newspaper text; compare section 5. The threshold value 0.3 has been used for all languages and in all experiments that we describe in section 6.

4 Token-Based Classification

The second classification stage of Punkt operates on the token level and improves the intermediate annotation of the corpus provided by the type-based classification stage. For every token with a final period, the system decides on the basis of its immediate right context whether the intermediate annotation has to be modified or corrected. For this classification step, the relevant tokens are separated into different classes (cf. Figure 1). Each class triggers a re-examination with a different combination of a few basic heuristics. The most important classes are ordinary abbreviations and ellipses, which may appear at the end of a sentence. Moreover, there are two special classes of abbreviations, which are problematic for type-based approaches, namely possible initials and possible ordinal numbers.

Three basic heuristics are employed in the token-based classification stage: the orthographic heuristic, whose task is to test for orthographic clues for the detection of sentence boundaries after abbreviations and ellipses, the collocation heuristic, which determines whether two words surrounding a period form a collocation and interprets a positive answer to this question as evidence against an intervening sentence boundary, and finally the frequent sentence starter heuristic, which suggests a preceding sentence boundary if a word appearing after a period is found on a list of frequent sentence starters induced on the fly from the text that is to be segmented.

4.1 Heuristics in the Token-Based Stage

4.1.1 The Orthographic Heuristic At first sight, it might seem reasonable to rely on orthographic conventions for the detection of sentence boundaries. For instance, a capitalized word usually indicates a preceding sentence boundary in mixed-case text. However, such a procedure is perilous for various reasons. To begin with, certain word classes are capitalized even if they occur sentence-internally as is the case with the majority of proper nouns in English and all nouns in German. Even a lower case first letter does not guarantee that the word in question is not preceded by a sentence boundary. This is particularly evident for mathematical variables or names that are conventionally written without capitalization such as *amnesty international*; see also Nunberg (1990, pages 54f.). Finally, any method that relies solely on capitalization will not help at all with single-case text.

Still, we think that capitalization information – if used cautiously – can help to determine whether an abbreviation or ellipsis precedes a sentence boundary or not; see section 6.4.5 for a discussion of the importance of orthographic information in comparison to other types of evidence. In order to make the usage of orthographic information safer and more robust, Punkt counts how often every word type occurs with an uppercase and lowercase first letter at the beginning of a sentence and sentence-internally in the corpus; see Table 3 for some example statistics. It bases its calculations on the sentence boundaries determined by the type-based classification stage. It does not count tokens occurring after an abbreviation or an ellipsis, because we have not yet deter-

Figure 3
Pseudo Code of the Orthographic 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
```

mined whether they start a new sentence or not. The algorithm also ignores tokens that occur after possible initials and numbers. Again, a re-classification is likely to happen; see section 4.3. In sum, as the counts are based on imperfectly annotated data, we try to exclude most doubtful cases.

Figure 3 gives a pseudo code description of the orthographic heuristic, which decides for a token following an abbreviation or an ellipsis on the basis of the orthographic statistics gathered for all word types whether it represents good evidence for a preceding sentence boundary or not.

If the word following an abbreviation or ellipsis is capitalized, the heuristic determines whether it occurs with a lowercase first letter in the text and whether it occurs with an uppercase first letter sentence-internally. Only if it occurs with a lowercase first letter at least once and never occurs with an uppercase first letter sentence-internally, the heuristic opts for a sentence boundary after the abbreviation or ellipsis. Otherwise, it returns *undecided*.

If the token following the abbreviation or ellipsis has a lowercase first letter, the heuristic decides against a sentence boundary if that type also occurs with an uppercase first letter or if it never occurs with a lowercase first letter after a sentence boundary. In all other cases, the heuristic returns *undecided*.

Table 3
Collected Data for the Orthographic Heuristic

Type	Uppercase All	Lowercase All	Uppercase after Sure Sentence Boundary	Lowercase after Sure Sentence Boundary	Uppercase Clearly Sentence Internally	Lowercase Clearly Sentence Internally
a	2,229	34,483	720	0	654	34,466
across	2	129	1	0	0	129
actual	3	52	2	0	0	52
ask	9	140	1	0	7	140
psychologists	2	4	1	0	0	4
smith	348	0	6	0	218	0

For example, if the input of the orthographic heuristic is the capitalized token *A*, the heuristic would return *undecided* given the data in Table 3, because the type *a* occurs capitalized both at the beginning of and inside a sentence. If the input is the lowercase token *a*, it would return *no sentence boundary*. For the proper name *Smith*, the result would be *undecided* since the name never occurs with a lowercase first letter at all. For the input tokens *Across*, *Actual*, and *Psychologists*, the heuristic would decide in favor of a *sentence boundary*. If the same tokens were not capitalized, the decision would be *no sentence boundary*.

In the worst case scenario of an all-uppercase corpus, the orthographic heuristic would always return *undecided*. If all tokens in the corpus begin with a lowercase letter, it would either return *undecided* or *no sentence boundary*, which we think is the safest option since in this case, the heuristic cannot adduce any additional token-based evidence and most sentence boundaries have already been discovered by the type-based stage. The option to refuse a decision on the basis of capitalization information thus makes the orthographic heuristic very robust.

4.1.2 The Collocation Heuristic The basic intuition behind the collocation heuristic is that sentence boundaries block collocational ties (Manning and Schütze, 1999, page 195). If a period is surrounded by two words that form a collocation, we do not expect it to act as a sentence boundary marker. A period should therefore be interpreted as an abbreviation marker and not as a sentence boundary marker if the two tokens surrounding it can indeed be considered as a collocation according to Dunning's (1993) original log-likelihood ratio amended with the one-sidedness constraint introduced in section 2.2. Following the asymptotic χ^2 distribution of Dunning's original proposal, we require a log λ value of at least 7.88, which represents a confidence of 99.5 %. If this condition is met, we assume that the two words surrounding the potential sentence boundary form a collocation and hence represent evidence against an intervening sentence boundary.

4.1.3 The Frequent Sentence Starter Heuristic We also employ Dunning's log λ as an unsupervised method for the extraction of frequent sentence starters, that is, word types that occur particularly often after a sentence boundary. We take the viewpoint of collocation detection and define a frequent sentence starter as a word type that has a strong collocational tie to a preceding sentence boundary. The occurrence of such a frequent sentence starter after a period can thus be used to adduce further evidence that the preceding period marks a sentence boundary.

In contrast to Mikheev (2002, page 297), we do not extract the list of frequent sentence starters from an additional corpus but use the test corpus itself. The basic idea is to build a list of frequent sentence starters on the fly by counting how often every word type occurs following a sure sentence boundary, as determined by the type-based first stage. Sure sentence boundaries are all single periods following words that have been classified as non-abbreviations and are not possibly initials or numbers, that is, single letters or digits. Once the problem has been formulated as a problem of collocation detection, we can use the amended version of Dunning's log λ, as described in section 2.2, to test the candidate word types for a collocational tie to a preceding sentence boundary. Since we are relying on uncertain information, namely the intermediate annotation, we assume an exceptionally high threshold value of 30 for the classification. Only if this value is reached or exceeded, we put the candidate on the list of frequent sentence starters. The high cut-off value has been determined experimentally during the development of Punkt (cf. section 5).

Finally, there exists an interaction between the frequent sentence starter heuristic and the collocation heuristic described in the preceding section in that the frequent sentence starter heuristic may help to counterbalance the collocation heuristic. Sometimes, the collocation heuristic will detect a collocation across a sentence boundary, particularly if the word preceding the boundary occurs quite often at the end of sentence and the word following the sentence boundary is a frequent sentence starter. By determining the frequent sentence starters in the corpus and preventing the detection of collocations with these types as second elements, collocation detection is made safer.

4.2 Token-Based Reclassification of Abbreviations and Ellipses

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 or not. A sentence boundary after these two classes of candidate tokens is assumed by Punkt if the *orthographic heuristic* applied to the token following the abbreviation or ellipsis decides in favor of a sentence boundary or the token following the abbreviation or ellipsis 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, as will be discussed in the following section.

4.3 Token-Based Detection of Initials and Ordinal Numbers

Initials are a subclass of abbreviations consisting of a *single* letter followed by a period.[5] As there are only about thirty different letters in the average Latin-derived alphabet, the likelihood of being a homograph of an ordinary word is very high for initials, consider, for example, the Portuguese definite articles *o* and *a* or the Swedish preposition *i* ('in'). Moreover, there are also various other uses for single letters: in formulas, enumerations, and so on. Initials are therefore often not detected by the type-based first stage of the Punkt system. For this reason, all single letters followed by a token-final period are treated as possible initials during the token-based reclassification – regardless of whether they have been classified as abbreviations or not by the type-based stage.

Luckily, initials are very often part of a complex name and can often 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 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, we employ a special heuristic for initials: If the orthographic heuristic returns *undecided* and the type following the possible initial always occurs with an uppercase first letter, it is assumed to be a proper name and the period between the two tokens is again classified as an abbreviation marker. The system never reclassifies a period following a possible initial as a sentence boundary marker because we assume that if a single letter is indeed not used as an abbreviation but for example as a mathematical symbol or if it is an ordinary word, there will usually be enough occurrences of this type without a final period so that the type-based stage will classify all periods following instances of the type in question as sentence boundary periods.

In many languages, such as German, ordinal numbers written in digits are also marked by a token-final period; compare example (2).

5 Sometimes, two or more initials are not separated from each other by spaces, compare *L.F. Rothschild* in example (1). Although these cases are usually still regarded as initials, our system does not treat them as such but as ordinary abbreviations because it cannot distinguish abbreviations with internal periods from such run-on combinations of initials. This is however not harmful because combinations like *L.F.* are normally short, contain internal periods, and will probably not occur without a following period so that there is a high likelihood that they will be recognized as abbreviations in the type-based stage.

Example 2
Was sind die Konsequenzen der Abstimmung vom ***12.*** *Juni?*
"What are the consequences of the poll of the 12^{th} of June?"
(cited from NZZ 06/13/1994)

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 marker 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*##[6] and is not a frequent sentence starter, the period in between is classified as an abbreviation period. The same conclusion is reached if the orthographic heuristic decides against a sentence boundary on the basis of the following token.

5 Development of the Punkt System

The three scaling factors used to improve on the initial sorting of the collocational criterion for abbreviation detection were obtained by manual experiments on a 10 MB development corpus of American English containing *Wall Street Journal* articles. This corpus is distinct from the portions of the WSJ we use for evaluation purposes in section 6. From this development corpus, a candidate list of possible abbreviation types was extracted and sorted according to the log-likelihood ratio described in section 2.1. We experimented with different factors and measured their impact in terms of precision and recall on the candidates above a given threshold value. Our goal was to maximize precision and recall for the top part of the list, that is, to get a clear separation of true abbreviations at the top of the list from non-abbreviations at the bottom of the list. The factors F_{length} (9) and $F_{periods}$ (10) were conceived and tested solely on the basis of the WSJ development corpus. We have added the third factor $F_{penalty}$ (11) to cope with the problem of very common ordinary words that precede a sentence boundary most of the time. We encountered this problem in the Turkish test corpus, but it would probably arise for other verb-final languages as well. After fixing the final form of the scaling factors, we also used the candidate list from the development corpus to determine the ideal threshold value for type-based abbreviation detection. The best combination of the different methods in the token-based stage and the threshold value 30 used for finding frequent sentence starters were also determined by manual experimentation on the development corpus. The same parameters and combinations of heuristics have been employed in all tests described in section 6 unless otherwise noted.

6 Evaluation

We have tested our system extensively for a number of different languages and under different circumstances. We report the results that we obtained from our experiments in section 6.4, after giving a short characterization of the test corpora on which we did our evaluation in section 6.1, defining the performance measures we use in section 6.2 and proposing three baselines as lower bounds and standards of comparison in section 6.3. We compare our approach to other systems for sentence boundary detection proposed in the literature in section 7.

6 As specific numbers often occur very infrequently, we fold all numeric types into one abstract type ##*number*## for the purposes of collocation detection.

Table 4
Newspaper Corpora Used in the Evaluation

Language	Origin	Content
Brazilian Portuguese	CETENFolha corpus (Linguateca)	Folha de S. Paulo
Dutch	Multilingual Corpus 1 (ECI)	De Limburger
English	Penn Treebank (LDC)	Wall Street Journal
Estonian	By courtesy of the University of Tartu	Eesti Ekspress
French	Multilingual Corpus 1 (ECI)	Le Monde
German	Neue Zürcher Zeitung AG CD-ROM	Neue Zürcher Zeitung
Italian	Multilingual Corpus 1 (ECI)	La Stampa, Il Mattino
Norwegian	By courtesy of the Centre for Humanities Information Technologies, Bergen	Bergens Tidende (Bokmål and Nynorsk)
Spanish	Multilingual Corpus 1 (ECI)	Sur
Swedish	Multilingual Corpus 1 (ECI)	Dagens Nyheter (and others)
Turkish	METU Turkish Corpus (Türkçe Derlem Projesi), by courtesy of the University of Ankara	Milliyet

6.1 Test Corpora

We have evaluated Punkt on corpora from eleven different languages: Brazilian Portuguese, Dutch, English, Estonian, French, German, Italian, Norwegian, Spanish, Swedish, and Turkish. For all of these languages, we have created test corpora containing newspaper text, the genre that is most often used to test sentence boundary detection systems; compare section 7. Table 4 provides a short description for each one of these newspaper corpora. Five of them – Dutch, French, Italian, Spanish, and Swedish – are parts of corpora taken from the CD-ROM Multilingual Corpus 1 distributed by the European Corpus Initiative (ECI). For English, we have used sections 03-06 of the *Wall Street Journal* portion of the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993) distributed by the Linguistic Data Consortium (LDC), which have frequently been used to evaluate sentence boundary detection systems before; compare section 7.

For the other languages, we have chosen newspaper corpora that were either available on the internet (Brazilian Portuguese), distributed by the newspaper itself (German), or kindly provided to us by other research institutions (Estonian, Norwegian, and Turkish). While the Swedish corpus contains a small amount of literary fiction in addition to newspaper articles from several Swedish newspapers, the other corpora consist solely of newswire text.

Table 5
Other Corpora Used in the Evaluation

Language	Origin	Content
English	Brown Corpus	Balanced corpus of American English, different text genres
English	Project Gutenberg	The Works of Edgar Allan Poe in Five Volumes, Volumes I-III, literary fiction

Table 6
Tags Used in the Evaluation

<S>	Sentence Boundary
<A>	Abbreviation
<E>	Ellipsis
<A><S>	Abbreviation at the End of Sentence
<E><S>	Ellipsis at the End of Sentence

To determine whether Punkt is also suitable for different text genres, we have additionally evaluated it on a piece of American English literature; compare Table 5; obtained from Project Gutenberg: http://www.gutenberg.org/. Last but not least, we have also tested it on the Brown corpus of American English (Francis and Kucera, 1982), which has often been used to evaluate other sentence boundary disambiguation systems. This corpus contains a mixture of different text genres including newspaper text, scientific articles, and literary fiction.

For the evaluation, we have created annotated versions of the test corpora, in which all periods were disambiguated by hand and labeled with the correct tag from Table 6. Table 7 illustrates some statistical properties of the test corpora. For each corpus, we provide the number of tokens it contains and the number of all tokens with a final period, that is, all periods that had to be classified by Punkt. As an indication of the difficulty of the sentence boundary detection task, we also give information on how many abbreviations each corpus contains and what percentage of all the tokens with a final period actually are abbreviations. Finally, the last column shows the number of different abbreviation types occurring in each test corpus.

6.2 Performance Measures

The most important performance measure we use is the error rate, as given in (13). It is simply defined as the ratio of the number of incorrectly classified candidates to the number of all classified candidates.

Table 7
Statistical Properties of the Test Corpora

Corpus	Tokens	Tokens with Final Periods	Abbr. Tokens	Abbr. Tokens %	Abbr. Types
B. Portuguese	321,032	15,250	481	3.15 %	102
Dutch	340,238	20,075	1,270	6.33 %	141
English – WSJ	469,396	26,980	7,297	27.05 %	196
English – Brown	1,105,348	54,722	5,586	10.21 %	213
English – Poe	324,247	11,247	600	5.33 %	59
Estonian	358,894	25,825	2,517	9.75 %	248
French	369,506	12,890	375	2.91 %	91
German	847,207	38,062	3,603	9.47 %	139
Italian	312,398	11,561	442	3.82 %	156
Norwegian	479,225	28,368	1,882	6.63 %	242
Spanish	352,773	13,015	570	4.38 %	84
Swedish	338,948	19,724	769	3.90 %	100
Turkish	333,451	21,047	598	2.84 %	103

$$\text{error rate} = \frac{\text{false positives} + \text{false negatives}}{\text{number of all candidates}} \tag{13}$$

In addition to the error rate, we use precision and recall to provide better information on what kinds of errors were made by Punkt. Precision is the ratio between the number of candidate tokens that have been correctly assigned to a class and the number of all candidates that have been assigned to this class.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{14}$$

Recall is defined as the proportion of all candidates truly belonging to a certain class that have also been assigned to that class by the evaluated system.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{15}$$

Finally, the so-called F measure is the harmonic mean of precision and recall (van Rijsbergen, 1979).

$$\text{F measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{16}$$

All the measures we use are based on counting true and false positives and true and false negatives. There are, however, three possibilities of what classifications could be regarded as a positive or a negative outcome because Punkt actually performs three classification tasks at the same time. The most important one is the decision whether a token-final period marks a sentence boundary or not. But the system also decides for each candidate token ending in a period whether it is an abbreviation or not and whether it is an ellipsis or not; compare Table 6. The performance measures for sentence boundary detection, abbreviation detection, and ellipsis detection do not directly depend on each other because a candidate can be an abbreviation (or an ellipsis) and precede a sentence boundary at the same time. We thus calculate error rate, precision, recall, and F measure for the sentence boundary detection problem and for the abbreviation detection problem separately. For sentence boundary detection, we count the following tags as positives: <S>, <A><S>, and <E><S>. The tags <A> and <A><S> are considered positives for the abbreviation detection task. As the correct classification of ellipses is straightforward, we do not give figures for the detection of ellipses.

6.3 Baselines

In the evaluation in sections 6.4.1 and 6.4.2, we employ three different baseline algorithms as standard for comparison and evaluate Punkt against these standards. These baselines serve several purposes. First, they establish a lower bound for the task of sentence boundary detection. Any sentence boundary detection system should perform significantly better than these baseline algorithms. Second, although we compare Punkt to other systems proposed in the literature in section 7, most previous work on sentence boundary detection considered at most three different languages so that no direct comparison is possible for many of the corpora and languages that we have used in our evaluation. A comparison with the performance of the three baselines can at least give an indication of how well our system did on these corpora. Third, there is still an assumption held in the field that simple algorithms such as the baselines presented here are sufficiently reliable to be used for sentence boundary detection. This opinion was, for example, held by a reviewer of Kiss and Strunk (2002a). As will become clear in the following sections, a baseline algorithm may perform pretty well on one corpus, but this performance typically does not carry over to other languages or corpora. The baselines thus also serve to illustrate the complexity of the sentence boundary detection problem.

The *absolute baseline* (AbsBL) is the simplest approach to sentence boundary detection we can think of. It simply assumes that all token-final periods in a test corpus represent sentence boundaries. Consequently, all periods are annotated with the tag <S>.

The second baseline algorithm (TokBL) relies only on the local context of the period it classifies. It is a token-based approach that uses only orthographic information. All token-final periods (including those that form part of an ellipsis) that do not precede a token starting with a lowercase letter, a digit, or one of the following sentence internal punctuation marks [; : ,] are classified as sentence boundary markers and annotated with <S>. All other token-final periods are either classified as abbreviations (<A>) or ellipses (<E>).

The third baseline algorithm (TypeBL) is based on a method described by Grefenstette (1999) and also by Mikheev (2002, page 299). It is a type-based approach, that is, the algorithm decides for each candidate type whether it is an abbreviation or not. All instances of candidates that ever occur in an unambiguous position, that is, before a lowercase letter or a sentence-internal punctuation mark [; : ,] are classified as abbreviations and a period following them is not considered as an end-of-sentence marker. All other candidate types are treated as ordinary words and a period following them is classified as a sentence boundary marker. No sentence boundary is assumed after ellipses.

6.4 Experiments

We have tested Punkt on the various corpora introduced in section 6.1. In all cases, it was only provided with the unannotated test corpus as input and no further data whatsoever, most importantly no lexicon and no list of abbreviations. Its main classification task was to decide for all token-final periods whether they indicated the end of a sentence or not.[7]

Table 8
Results of Classification – Newspaper Corpora (Mixed Case)

Corpus	**Error (<S>)**	**Prec. (<S>)**	**Recall (<S>)**	**F (<S>)**	**Error (<A>)**	**Prec. (<A>)**	**Recall (<A>)**	**F (<A>)**
B. Port.	1.11 %	99.14 %	99.72 %	99.43 %	0.99 %	96.88 %	70.89 %	81.87 %
Dutch	0.97 %	99.25 %	99.72 %	99.48 %	0.66 %	99.31 %	90.24 %	94.55 %
English	1.65 %	99.13 %	98.64 %	98.89 %	0.71 %	99.86 %	97.52 %	98.68 %
Estonian	2.12 %	98.58 %	99.07 %	98.83 %	1.75 %	98.22 %	83.51 %	90.27 %
French	1.54 %	99.31 %	99.08 %	99.19 %	0.72 %	95.19 %	79.20 %	86.46 %
German	0.35 %	99.69 %	99.93 %	99.81 %	0.26 %	99.91 %	97.34 %	98.61 %
Italian	1.13 %	99.32 %	99.49 %	99.41 %	0.74 %	96.60 %	83.48 %	89.56 %
Norw.	0.81 %	99.45 %	99.68 %	99.56 %	0.72 %	98.16 %	90.81 %	94.34 %
Spanish	1.06 %	99.66 %	99.23 %	99.45 %	0.35 %	98.70 %	93.33 %	95.94 %
Swedish	1.76 %	98.82 %	99.36 %	99.09 %	1.48 %	94.10 %	66.32 %	77.80 %
Turkish	1.31 %	99.40 %	99.24 %	99.32 %	0.43 %	95.35 %	89.13 %	92.13 %
Mean	1.26 %	99.25 %	99.38 %	99.31 %	0.80 %	97.48 %	85.62 %	90.93 %
Std. Dev.	0.49 %	0.33 %	0.38 %	0.29 %	0.46 %	1.99 %	10.19 %	6.69 %

7 Only periods were classified. We did not include the less ambiguous exclamation and question marks in the evaluation.

Table 9
Comparison with the Baselines – Newspaper Corpora (Mixed Case)

	Error <S>				**Error <A>**			
Corpus	**Punkt**	**AbsBL**	**TokBL**	**TypeBL**	**Punkt**	**AbsBL**	**TokBL**	**TypeBL**
B. Port.	1.11 %	3.17 %	2.01 %	1.74 %	0.99 %	3.15 %	2.11 %	1.51 %
Dutch	0.97 %	6.50 %	5.66 %	1.63 %	0.66 %	6.33 %	5.59 %	1.36 %
English	1.65 %	25.60 %	13.37 %	7.14 %	0.71 %	27.05 %	14.96 %	5.40 %
Estonian	2.12 %	10.03 %	4.86 %	7.45 %	1.75 %	9.75 %	4.94 %	6.53 %
French	1.54 %	4.20 %	3.02 %	2.87 %	0.72 %	2.91 %	2.20 %	1.19 %
German	0.35 %	9.50 %	6.23 %	8.74 %	0.26 %	9.47 %	6.22 %	8.60 %
Italian	1.13 %	4.45 %	3.40 %	3.14 %	0.74 %	3.82 %	3.11 %	2.53 %
Norw.	0.81 %	6.57 %	2.98 %	5.44 %	0.72 %	6.63 %	3.09 %	5.10 %
Spanish	1.06 %	4.23 %	3.17 %	2.61 %	0.35 %	4.38 %	3.40 %	1.75 %
Swedish	1.76 %	4.02 %	1.68 %	2.58 %	1.48 %	3.90 %	1.83 %	1.79 %
Turkish	1.31 %	3.47 %	5.25 %	26.40 %	0.43 %	2.84 %	4.66 %	25.44 %
Mean	1.26 %	7.43 %	4.69 %	6.23 %	0.80 %	7.29 %	4.74 %	5.56 %
Std. Dev.	0.49 %	6.46 %	3.24 %	7.48 %	0.46 %	7.01 %	3.69 %	7.05 %

In addition, it had to decide for all token-final periods whether they were used as abbreviation marker or were part of an ellipsis.

The results Punkt achieved on the newspaper corpora are presented in section 6.4.1. The results obtained for the remaining corpora are given in section 6.4.2. In section 6.4.3, we provide the results of an experiment in which we evaluated our system on all-uppercase and all-lowercase corpora. As many competing systems require a list of abbreviations, we have carried out an experiment to determine the usefulness of abbreviation lists derived from general purpose dictionaries. The results are reported in section 6.4.4. Last but not least, we take a closer look at the architecture of Punkt in section 6.4.5 by examining the contributions of its individual parts and discuss the hypothesis that the methods and heuristics we employ can be called language independent in section 6.4.6.

6.4.1 Results on the Newspaper Corpora Table 8 shows the results that we obtained for the tasks of sentence boundary detection and abbreviation detection on the eleven newspaper corpora. We performed two test runs for each language: one with detection of ordinal numbers and one without a special treatment of numbers. For languages such as English, which do not usually mark ordinal numbers with a final period, it is obviously preferable not to try to detect them. In Table 8, we only report the best result from the two test runs for each language.[8] Those languages in which the period is not usually used to mark ordinal numbers and for which the test without special treatment of numbers achieved better results are italicized in the following tables. However, even if the special treatment of numbers was not turned off for such languages, the resulting increase in the error rate was not very high, maximally 0.03 %; see also section 6.4.5.

For the sentence boundary detection task, the error rates Punkt achieved on the eleven newspaper corpora range from 2.12 % on the Estonian corpus to only 0.35 % on

8 One exception is the French corpus. It contains rankings from sports events in which ranks are indicated using digits and a following period. Using the special detection of ordinal numbers on this corpus results in a lower error rate of 1.33 % for the task of sentence boundary detection and 0.50 % for abbreviation detection. However, as ordinal numbers in French are not usually indicated with final periods, we have given the results of the system without special treatment of ordinal numbers for French in Table 8.

Table 10
Results of Classification – Other Corpora (Mixed Case)

Corpus	Error (<S>)	Prec. (<S>)	Recall (<S>)	F (<S>)	Error (<A>)	Prec. (<A>)	Recall (<A>)	F (<A>)
Brown	1.02 %	99.14 %	99.75 %	99.44 %	0.82 %	98.92 %	92.17 %	95.43 %
Poe	0.80 %	99.71 %	99.45 %	99.58 %	0.46 %	95.36 %	96.00 %	95.68 %

the German corpus with an average error rate of 1.26 %. The error rates for the abbreviation detection task are slightly lower, lying between 1.75 % on the Estonian corpus and 0.26 % on the German corpus with an average of about 0.80 % for all eleven corpora.

Table 9 compares Punkt's performance to that of the three baseline algorithms. The error rates achieved by Punkt for the sentence boundary task are reduced by about 83 % percent on average compared to the absolute baseline, by about 73 % compared to the token-based baseline, and by almost 80 % compared to the type-based baseline. The error rates for the abbreviation detection task have decreased even more considerably, namely by approximately 89 % in comparison to the absolute baseline, by about 83 % in comparison to the token-based baseline, and by almost 86 % in comparison to the type-based baseline. Table 9 also shows that whereas the good performance of our system is quite stable across the eleven corpora with a standard deviation of only 0.49 % for sentence boundary detection and 0.46 % for abbreviation detection, the performance of the baselines is not reliable at all. Although one of the baselines sometimes performed well on one corpus – such as TokBL on the Swedish corpus, the only case where Punkt was not better than all of the baselines, or TypeBL on the Brazilian Portuguese and Dutch corpora, the baselines exhibit a very large standard deviation in their error rates across the eleven corpora and sometimes seem to fail completely, such as TokBL on the English corpus and TypeBL on the Turkish corpus.

6.4.2 Results on the Other Corpora In order to show that Punkt is also suited to process text genres different from newspaper text and that its performance carries over to other text types, we have tested it on two additional corpora of American English – the entire Brown corpus and The Works of Edgar Allan Poe (volumes I–III). Table 10 provides the results Punkt achieved on the two additional corpora. The error rate on the Brown corpus is 1.02 % for the sentence boundary detection task and 0.82 % for the abbreviation detection task. This represents a reduction of about 90 % compared to the absolute baseline, a reduction of more than 85 % compared to the token-based baseline, and a reduction of more than 70 % compared to the type-based baseline; compare Table 11. The error rate on the Works of Edgar Allen Poe is 0.80 % for sentence boundary detection and 0.46 % for abbreviation detection, which corresponds to a reduction of about 85 % in comparison to AbsBL, a reduction by more than 80 % compared to TokBL and by about 75 % compared to TypeBL. These results achieved by Punkt on the literary Poe corpus

Table 11
Comparison with the Baselines – Other Corpora (Mixed Case)

	Error <S>				Error <A>			
Corpus	**Punkt**	**AbsBL**	**TokBL**	**TypeBL**	**Punkt**	**AbsBL**	**TokBL**	**TypeBL**
Brown	1.02 %	9.83 %	7.17 %	3.59 %	0.82 %	10.21 %	7.55 %	3.27 %
Poe	0.80 %	5.03 %	4.12 %	3.12 %	0.46 %	5.33 %	4.42 %	2.76 %

and the Brown corpus with its balanced content fall within the range of the error rates achieved on the newspaper corpora and thus indicate that Punkt is also well-suited to deal with literary texts and corpora containing mixed content.

6.4.3 Experiments with Single-Case Corpora We have also tested the applicability of Punkt to single-case text. The newspaper test corpora have been converted to all-uppercase and all-lowercase versions in order to determine how much Punkt is affected by the loss of capitalization information. In fact, one should keep in mind that single-case text does not only lack useful capitalization information, but actually contains information that is highly misleading for systems that rely primarily on capitalization. Table 12 shows the performance of our system on the single-case corpora. The left half of the table contains the error rates and F values for sentence boundary detection and abbreviation detection on the lowercase corpora. The right half gives the corresponding values for the tests on the uppercase corpora. The last two rows of the table compare these results with those Punkt achieved on the mixed-case versions (MC) of the newspaper corpora; compare section 6.4.1. As can be seen in Table 12, the performance of our system is only minimally affected by the loss of capitalization information, slightly more so on the all-lowercase corpora. The error rate our system produces for the task of sentence boundary detection is 0.41 % higher on average on the lowercase corpora than on the mixed-case corpora. The increase in the error rate on the uppercase corpora is slightly lower: 0.29 %. For the task of abbreviation detection, the increase in the error rates is even lower: 0.14 % on the lowercase corpora and 0.13 % on the uppercase corpora. This is expected because Punkt only uses capitalization information as evidence during the token-based correction and reclassification stage and not as primary evidence for the detection of abbreviations. The experiments on the single-case corpora show that Punkt is quite robust and well-suited also to process single-case text.

6.4.4 Experiments with Additional Abbreviation Lists Punkt is able to dynamically detect abbreviations in the test corpus itself. It therefore does not depend on precom-

Table 12
Results of Classification – Newspaper Corpora (Single Case)

	All-Lowercase Corpora				All-Uppercase Corpora			
Corpus	**Error (<S>)**	**F (<S>)**	**Error (<A>)**	**F (<A>)**	**Error (<S>)**	**F (<S>)**	**Error (<A>)**	**F (<A>)**
B. Port.	1.25 %	99.36 %	1.11 %	82.68 %	1.19 %	99.39 %	1.04 %	81.28 %
Dutch	1.00 %	99.47 %	0.72 %	94.12 %	0.85 %	99.55 %	0.57 %	95.34 %
English	2.30 %	98.44 %	0.71 %	98.66 %	2.04 %	98.63 %	0.72 %	98.65 %
Estonian	2.57 %	98.57 %	1.66 %	91.01 %	2.80 %	98.45 %	2.10 %	88.13 %
French	2.56 %	98.65 %	0.85 %	84.36 %	1.99 %	98.96 %	0.85 %	84.38 %
German	0.40 %	99.78 %	0.26 %	98.62 %	0.47 %	99.74 %	0.30 %	98.38 %
Italian	1.48 %	99.23 %	0.83 %	88.38 %	1.37 %	99.29 %	0.80 %	88.97 %
Norwegian	1.55 %	99.17 %	1.32 %	89.73 %	1.41 %	99.25 %	1.18 %	90.54 %
Spanish	1.31 %	99.31 %	0.45 %	94.69 %	1.12 %	99.41 %	0.32 %	96.24 %
Swedish	2.39 %	98.76 %	1.86 %	72.88 %	2.28 %	98.82 %	1.74 %	74.25 %
Turkish	1.53 %	99.20 %	0.57 %	89.74 %	1.54 %	99.20 %	0.58 %	89.30 %
Mean	1.67 %	99.09 %	0.94 %	89.53 %	1.55 %	99.15 %	0.93 %	89.59 %
Std. Dev.	0.70 %	0.42 %	0.50 %	7.54 %	0.67 %	0.40 %	0.56 %	7.56 %
Mean (MC)	1.26 %	99.31 %	0.80 %	90.93 %	1.26 %	99.31 %	0.80 %	90.93 %
Difference	0.41 %	-0.22 %	0.14 %	-1.40 %	0.29 %	-0.16 %	0.13 %	-1.34 %

Table 13
Abbreviation Lists

List	English	German
All Abbreviations	1537	769
No Homographs	1115	729
No Single Characters	1513	742
No Homographs and No Single Characters	1112	703

piled abbreviation lists like some of its competitors; compare section 7. But even though an abbreviation list is not necessary for Punkt to perform well, such a list can easily be integrated into its architecture. The abbreviations read from such a list are simply added to those the system has detected in the test corpus after the type-based stage. Ideally, one would use a domain-specific abbreviation list if the domain of the test corpus is known beforehand. However, we wanted to determine the usefulness of general-purpose abbreviation lists derived from general-purpose dictionaries. We have therefore built such abbreviation lists by extracting by hand all abbreviations from a German spelling dictionary – the Rechtschreibduden (Dudenredaktion, 2004) – and all English abbreviations from a bilingual dictionary – the small Muret-Sanders English-German dictionary by Langenscheidt (Willmann and Messinger, 1996). This yielded a total number of 769 abbreviations for German and 1,537 for English; compare Table 13.

We then made three additional versions of these lists from which we deleted potentially harmful entries: one from which we removed all abbreviations that had obvious non-abbreviation homographs, one from which we removed all single character abbreviations, and one from which we removed both abbreviations homographic to ordinary words and single character abbreviations. Table 13 gives the number of remaining abbreviations for these different versions. We produced the additional versions of the lists in order to test how much care is needed when preparing abbreviation lists for a sentence boundary detection system like ours.

We then carried out two experiments on the German and English newspaper corpora and the Brown corpus. In the first experiment, we tested how well Punkt performed when it was provided with the different abbreviation lists in addition to the abbreviations it was able to detect on the fly. Table 14 contains the results of this experiment. They show that Punkt can indeed benefit from additional abbreviation lists, but only if these are prepared with care by excluding abbreviations homographic to or-

Table 14
Results of Classification – Using an Additional Abbreviation List

Error <S>					
Corpus	**No List**	**All**	**No Homogr.**	**No Single Chars.**	**Neither**
WSJ	1.65 %	1.97 %	1.58 %	1.96 %	1.58 %
Brown	1.02 %	1.75 %	0.93 %	1.72 %	0.92 %
NZZ	0.35 %	0.37 %	0.32 %	0.37 %	0.32 %

Error <A>					
Corpus	**No List**	**All**	**No Homogr.**	**No Single Chars.**	**Neither**
WSJ	0.71 %	0.89 %	0.63 %	0.88 %	0.63 %
Brown	0.82 %	1.44 %	0.69 %	1.42 %	0.70 %
NZZ	0.26 %	0.28 %	0.23 %	0.28 %	0.23 %

Table 15
Results of Classification – Using Only a Fixed Abbreviation List

Error <S>					
Corpus	**On the Fly**	**All**	**No Homogr.**	**No Single Chars.**	**Neither**
WSJ	1.65 %	5.33 %	16.62 %	5.35 %	16.65 %
Brown	1.02 %	2.53 %	5.24 %	2.60 %	5.26 %
NZZ	0.35 %	2.55 %	2.64 %	2.59 %	2.67 %

Error <A>					
Corpus	**On the Fly**	**All**	**No Homogr.**	**No Single Chars.**	**Neither**
WSJ	0.71 %	4.57 %	16.70 %	4.58 %	16.73 %
Brown	0.82 %	2.32 %	5.32 %	2.43 %	5.36 %
NZZ	0.26 %	2.48 %	2.56 %	2.51 %	2.60 %

dinary words and single character abbreviations. Providing such a carefully prepared abbreviation list reduced the error rate of our system on the WSJ corpus from 1.65 % to 1.58 %, the error rate on the Brown corpus from 1.02 % to 0.92 %, and the error rate on the German NZZ corpus from 0.35 % to 0.32 %. Additional general-purpose abbreviation lists thus do improve the performance of our system, but the decrease of the error rate is not very great. Table 14 also shows that the use of abbreviation lists from which abbreviations homographic to ordinary words have not been removed is not helpful at all and instead leads to an increased error rate on all of the three corpora.

In the second experiment, Punkt could only use the abbreviations on the different lists and was not allowed to add any additional abbreviations on the fly. This experiment thus really tests the coverage of general-purpose abbreviation lists and also the productivity of abbreviation use in the test corpora. Table 15 contains the results from this experiment. The column *On the Fly* in this table gives the error rates that Punkt achieved on the corpora in its normal configuration detecting abbreviations on the fly without being provided with an additional abbreviation list. The remaining columns show the results that Punkt produced when it could only use a fixed list of abbreviations and was not allowed to detect abbreviations on the fly. A comparison between the first column and the other columns makes clear that abbreviation use in the corpora is quite productive and that fixed general-purpose abbreviation lists are clearly not sufficient for sentence boundary detection. A versatile sentence boundary detection system should therefore always be able to detect unknown abbreviations on the fly.

6.4.5 Contributions of the Individual Parts of the System In this section, we take a look at the contributions of the individual parts of the system to its overall performance. First, we tried to determine the effectiveness of reclassifying the different candidate classes in

Table 16
Configurations for Testing the Effectiveness of Separate Reclassification in the Token-Based Stage

System	Description
1	Only type-based stage
2	Type-based stage + Reclassification of abbreviations (including initials) and ellipses
3	Type-based stage + Reclassification of abbreviations and ellipses + Separate treatment of initials
4	Type-based stage + Reclassification of abbreviations and ellipses + Separate treatment of initials + Detection of ordinal numbers (Complete System)

Table 17
Contributions of Separate Reclassifications – Newspaper Corpora (Mixed Case)

	Error <S>			
Corpus	**System 1**	**System 2**	**System 3**	**System 4**
B. Portuguese	1.37 %	1.27 %	1.11 %	1.12 %
Dutch	1.96 %	2.73 %	1.27 %	0.97 %
English	2.71 %	2.06 %	1.65 %	1.68 %
Estonian	7.37 %	6.88 %	6.46 %	2.12 %
French	2.94 %	2.04 %	1.54 %	1.33 %
German	2.38 %	2.32 %	2.25 %	0.35 %
Italian	2.18 %	1.90 %	1.13 %	1.14 %
Norwegian	4.09 %	3.92 %	3.38 %	0.81 %
Spanish	1.81 %	1.67 %	1.06 %	1.08 %
Swedish	2.42 %	2.16 %	1.95 %	1.76 %
Turkish	1.89 %	1.72 %	1.70 %	1.31 %

the token-based stage separately using specific combinations of evidence, namely ordinary abbreviations, ellipses, initials, and numbers. We therefore built four different versions of our system, which are described in Table 16. The different configurations become cumulatively more specialized in their treatment of the different candidate classes from *system 1* with no token-based reclassification at all to the complete *system 4*.

Table 17 gives the results that the four different configurations achieved on the eleven newspaper corpora. It shows that the cumulatively more specialized treatment of the different candidate classes helps to improve on the error rate of the type-based stage considerably. Moreover, a separate reclassification of initials and numbers is quite effective for reducing the error rate on the newspaper corpora, often even more so than the detection of sentence boundaries after abbreviations and ellipses. The separate treatment of initials is quite beneficial for the English corpus for example, reducing the error rate from 2.06 % to 1.65 % (*system 2* vs. *system 3*), but also for Dutch, French, Italian, Norwegian and Spanish, while the detection of ordinal numbers is a very important factor for the German newspaper corpus, reducing the error rate from 2.25 % to only 0.35 % (*system 3* vs. *system 4*), and also for the Estonian, Norwegian, and Turkish corpora. For all languages that use the period to mark ordinal numbers, the detection of ordinal numbers thus turns out to be a very important subtask of sentence boundary disambiguation. A comparison between *system 3* and *system 4* also shows that leaving the detection of ordinal numbers on for languages that do not mark them with a final period is not really harmful resulting in a maximal increase in the error rate of 0.03 %.

Table 18
Configurations for Testing the Effectiveness of the Heuristics in the Token-Based Stage

System	Description
A	Only type-based stage
B	Type-based stage + Collocation heuristic
C	Type-based stage + Collocation heuristic + Frequent sentence starter heuristic
D	Type-based stage + Collocation heuristic + Frequent sentence starter heuristic + Orthographic heuristic
E	Type-based stage + Collocation heuristic + Frequent sentence starter heuristic + Orthographic heuristic + Special orthographic heuristic for initials (Complete System)

In a second experiment, we have tested the usefulness of the different heuristics used during the token-based stage. Table 18 provides information on the five different configurations we have evaluated. We have again added heuristics cumulatively: first, the collocation heuristic, next the frequent sentence starter heuristic, then the orthographic heuristic, and finally the special orthographic heuristic for initials.

Table 19 contains the error rates produced by *Systems A* to *E* on the eleven newspaper corpora. It confirms that all heuristics contribute to the performance of the system, though to different degrees depending on the specific corpus. It also shows that the collocation heuristic is very effective in reducing the error rate on the different corpora, more effective in fact than the orthographic heuristic. This fact supports our argument that the importance of brittle orthographic evidence can be reduced and sentence boundary detection can be made more robust by relying more on collocational evidence. The collocation heuristic reduces the error rate from 7.37 % to 2.94 % on the Estonian corpus, for example, and is also very effective for German, Norwegian, and Spanish. The impact of the frequent sentence starter heuristic is somewhat smaller, but it still leads to a substantial decrease in the error rate from 2.61 % to 1.96 % for French and to smaller reductions for all other languages. Although Punkt does not rely so much on capitalization, the orthographic heuristic still reduces the error rate from 2.80 % to 2.18 % for Estonian, for example, and leads to smaller improvements for the other languages except for English, where it causes a small increase in the error rate. As most combinations of initials and a following proper name are already captured by the collocation heuristic, the special orthographic heuristic for initials is only applied to complex names that occur infrequently and thus does not result in a large reduction of the error rate. Still, it never has a negative effect and is able to reduce the error rate from 1.84 % to 1.65 % on the English corpus and from 1.78 % to 1.54 % on the French corpus. We conclude that our heuristics are well motivated in that they decrease the error rates on the newspaper corpora substantially and never have any severe detrimental effect. Moreover, as section 6.4.3 shows, they work effectively even for single-case corpora.

6.4.6 Language Independence and Optional Recalibration Punkt is conceived as a language and domain independent corpus preprocessing tool that can be used out of the box for all languages that use an alphabetic writing system and employ one and the same symbol to mark abbreviations and the end of sentence. It is our hypothesis that the thresholds we use in classification should not vary much from language to language

Table 19
Contributions of the Heuristics – Newspaper Corpora (Mixed Case)

	Error <S>				
Corpus	**System A**	**System B**	**System C**	**System D**	**System E**
B. Portuguese	1.37 %	1.28 %	1.13 %	1.12 %	1.11 %
Dutch	1.96 %	1.17 %	1.13 %	1.05 %	0.97 %
English	2.71 %	2.15 %	1.80 %	1.84 %	1.65 %
Estonian	7.37 %	2.94 %	2.80 %	2.18 %	2.12 %
French	2.94 %	2.61 %	1.96 %	1.78 %	1.54 %
German	2.38 %	0.47 %	0.42 %	0.36 %	0.35 %
Italian	2.18 %	1.60 %	1.39 %	1.23 %	1.13 %
Norwegian	4.09 %	1.44 %	1.34 %	0.87 %	0.81 %
Spanish	1.81 %	1.39 %	1.25 %	1.08 %	1.06 %
Swedish	2.42 %	2.13 %	1.94 %	1.76 %	1.76 %
Turkish	1.89 %	1.58 %	1.54 %	1.31 %	1.31 %

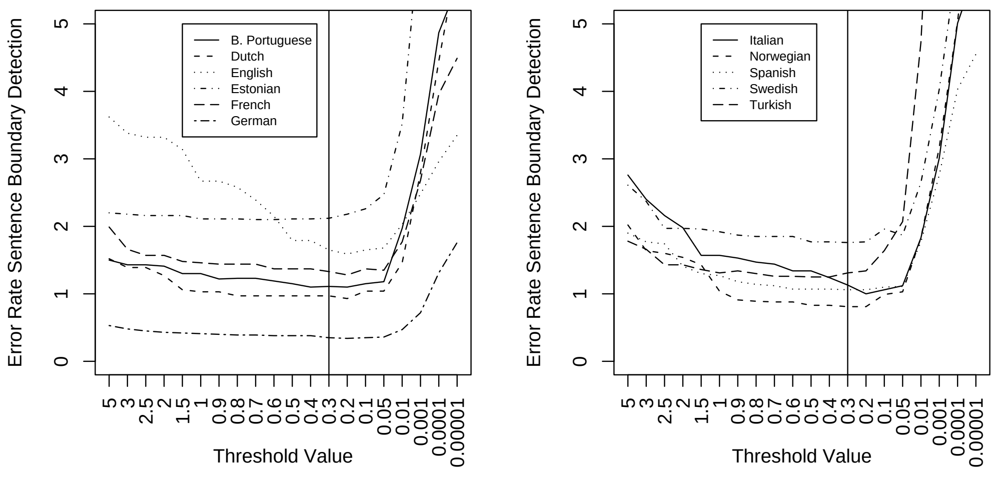

Figure 4
Error Rates for Different Classification Threshold Values in the Type-Based Stage

and from corpus to corpus. We have therefore determined optimal thresholds once on an English development corpus and retained these values for all experiments described so far; compare section 5. We have carried out two experiments in order to substantiate our hypothesis.

In the first experiment, we have tested different threshold values for the type-based abbreviation detection stage. Figure 4 shows that the threshold value of 0.3, which we have used so far, turns out to be the ideal value for three of the corpora and that the minimum error rate for all eleven corpora lies at or is very close to this threshold value. Table 20 gives the differences between the best error rate on each corpus and the error rate produced by our system with the chosen threshold value of 0.3. The maximal difference is 0.13 % for Italian. However, the average difference is only 0.03 %. The outcome of this experiment shows that abbreviations in the eleven languages behave very similarly and that the crucial type-based stage of Punkt can indeed be called language independent.

This is further corroborated by a second experiment in which we trained generalized linear models for the detection of abbreviation types on the eleven newspaper corpora using a logit link function and no intercept term. In this experiment, we used

Table 20
Difference between Lowest Error Rate and Error Rate achieved with a Threshold of 0.3

Language	0.3	Lowest	Diff.	Language	0.3	Lowest	Diff.
B. Portuguese	1.11 %	1.10 %	0.01 %	Italian	1.13 %	1.00 %	0.13 %
Dutch	0.97 %	0.93 %	0.04 %	Norwegian	0.81 %	0.81 %	0.00 %
English	1.65 %	1.59 %	0.06 %	Spanish	1.06 %	1.06 %	0.00 %
Estonian	2.12 %	2.10 %	0.02 %	Swedish	1.76 %	1.76 %	0.00 %
French	1.33 %	1.28 %	0.05 %	Turkish	1.31 %	1.25 %	0.06 %
German	0.35 %	0.34 %	0.01 %	Average Difference: 0.03 %			

the following three factors, which correspond to the collocational factor, the length factor, and the internal-periods factor used in the type-based stage of Punkt:

1. **Ratio**: The ratio of occurrences of a candidate with a final period to all occurrences of this candidate.
2. **Length**: The length of the candidate type (excluding periods).
3. **Periods**: The number of internal periods.

We then examined the resulting parameters of the trained models in order to determine whether the evidence we use in the type-based stage of Punkt is significant information for an effective model for abbreviation detection. All three factors always make a highly significant contribution and cannot be dropped from the models. The factor *Periods* is sometimes a little less important than the other two as there are some corpora in which most abbreviations do not contain internal periods. Figure 5 indicates the variation of the parameters that we obtained for the three factors (on the log-it link scale). It is remarkably small. The coefficient for the factor *Periods* exhibits the most variability, but is still quite stable. This relatively small variation of the parameters further substantiates our claim that the evidence we use for abbreviation detection can be considered language independent.

We know from further experiments that the threshold values 7.88 and 30, which we have chosen for the collocation heuristic and the frequent sentence starter heuristic, respectively, work well for our test corpora but are not always the optimal values for each individual corpus. Although Punkt is conceived as a flexible, unsupervised system, one can optionally recalibrate the threshold values by providing it with a hand-annotated training corpus. We have tested this possibility by annotating a second French corpus by hand. It again comprises articles from the newspaper *Le Monde*. It contains 371,526

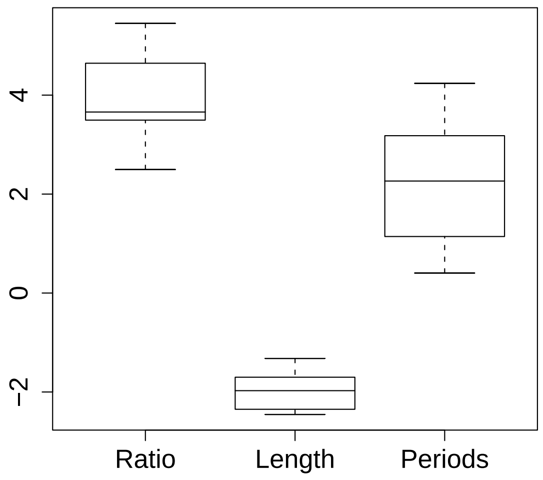

Figure 5
Variation of Parameters in a Log-Linear Model for Type-Based Abbreviation Detection

tokens in all and 13,664 tokens ending in a final period. Punkt achieved an error rate for sentence boundary detection of 1.84 % on this corpus. We used this second French corpus as training corpus to recalibrate the threshold values for the collocation heuristic and the frequent sentence starter heuristic. The optimal values determined on this training corpus were 17 for the collocation heuristic and 5 for the frequent sentence starter heuristic. We then used these values in a second test run on our original French test corpus. The resulting error rate for the task of sentence boundary detection was 1.44 %, while the error rate for abbreviation detection was 0.71 %. These results are a little better than the ones achieved without recalibration (1.54 % and 0.72 %); compare section 6.4.1. The optimal threshold values determined on the original test corpus itself are 6 for the collocation heuristic and 5 for the frequent sentence starter heuristic. The large difference between the best threshold values for the collocation heuristic on the two French corpora shows that the ideal value can vary from corpus to corpus, even if two corpora contain text of the same language and the same genre: in this case newspaper articles from *Le Monde*. Nevertheless, the lower error rate obtained in this experiment shows that Punkt can optionally be recalibrated on a training corpus to further optimize its performance. It can thus benefit from supervised training data, such as abbreviation lists or a training corpus but does not require such data in order to perform well.

7 Comparison to Other Systems

The results we presented in the previous section show that Punkt is able to achieve low error rates on corpora from eleven different languages, that it is well-suited to process different text genres, and that it is robust enough to deal with single-case text. Moreover, it reliably outperforms the three baseline algorithms and its performance is much more stable than that of the baselines.

In this section, we want to compare the performance of our system directly to that of competing systems and discuss advantages and disadvantages of the different approaches to sentence boundary detection. Unless otherwise indicated, the term *error rate* in this section always refers to the error rate for the task of sentence boundary detection.

7.1 Rule-Based Systems

Rule-based systems make use of hard-coded rules and fixed lists of lexical items such as abbreviations in order to identify which punctuation marks signal sentence boundaries and which do not, that is, they neither learn from an annotated training corpus nor use the test corpus itself to induce the required knowledge but rather employ precompiled resources usually provided by a human expert. We will discuss one such system by Silla Jr., Valle Jr., and Kaestner (2003) and compare its performance directly with that of our system. Moreover, we will also refer to results by Grefenstette (1999).

7.1.1 The RE system by Silla Jr., Valle Jr., and Kaestner (2003) The RE system[9] scans the test corpus until it encounters a period. It then compares the one token preceding and the one token following the period with a database of regular expressions that describe exceptions such as web addresses, decimal numbers, and most importantly abbreviations, in which the period does not indicate the end of a sentence. If the preceding token and/or the following token match a regular expression in the database, the RE system concludes that the period does not indicate a sentence boundary and searches

9 This is the name given to it in Silla Jr. and Kaestner (2004). *RE* stands for *regular expressions*. It is available from: http://www.ppgia.pucpr.br/∼silla/softwares/yasd.zip.

for the next period. If no matching regular expression is found, the period is classified as a sentence boundary.

In Strunk, Silla Jr., and Kaestner (2006), we have compared Punkt's performance to that of the RE system on two test collections: on articles from the *Wall Street Journal* taken from the TIPSTER document collection (TREC reference number: WSJ-910130) and on the Brazilian Portuguese Lacio-Web Corpus (Aluisio et al., 2003). The *RE system* was specifically developed for English newspaper texts. In order to use it on the Portuguese Lacio-Web corpus, 240 new regular expressions, which match Portuguese abbreviations, had to be added. Silla Jr. and Kaestner (2004) describe the adaptation process as "easy, although time consuming".

On the English TIPSTER test corpus, Punkt achieved results that were only slightly worse than those of the RE system: The RE system reached a precision of 92.39 % and a recall of 91.18 % which yielded an F measure of 91.78 %, while Punkt achieved a slightly lower precision of 90.70 % and a slightly higher recall of 92.34 % resulting in an F measure of 91.51 %. When comparing these results, it has to be kept in mind that the RE system employs a hand-crafted list of more than 700 abbreviations and was specifically developed for English newspaper text, while Punkt was not given any information besides the test corpus itself. Punkt's performance on the English TIPSTER corpus is thus quite impressive. This is further corroborated by the fact that Punkt was able to outperform the RE system on the Portuguese Lacio-Web corpus, even though 240 Portuguese abbreviations had been collected for the database of the RE system by hand: The RE system scored a precision of 91.80 % and a recall of 88.02 % which resulted in an F measure of 89.87 %, while Punkt achieved a precision of 97.58 % and a recall of 96.87 % yielding a much better F measure of 97.22 %. In sum, Punkt almost matched the performance of the RE system on the English test corpus and clearly outperformed it on the Portuguese test corpus.

7.1.2 Grefenstette (1999) – which is based on earlier work by Grefenstette and Tapanainen (1994) – discusses different approaches to sentence boundary disambiguation using different sources of information and evaluates them on the Brown corpus, on which Punkt achieved an error rate of 1.02 %. The first approach he describes is a simple regular expressions approach that tries to recognize abbreviations by matching against the following patterns (Grefenstette, 1999, page 127):

- a single capital followed by a period, such as "A.", "B." and "C.";
- a sequence of letter-period-letter-period's, such as "U.S.", "i.e." and "m.p.h.";
- a capital letter followed by a sequence of consonants followed by a period, such as "Mr.", "St." and "Assn.".

This approach produces a high error rate of 2.34 %. Moreover, it makes language-specific assumptions in that strings such as "Mr" might well be valid ordinary words in other languages. Grefenstette considers a second approach in which he additionally tries to identify abbreviations with a type-based heuristic similar to our type-based baseline (TypeBL) (Grefenstette, 1999, pages 128 and 129):

> Let us define as a *likely abbreviation* any string of letters terminated by a period and followed by either a comma or semi-colon, a question mark, a lower-case letter, or a number, or followed by a word beginning with a capital letter and ending in a period. [. . .] We can apply the corpus itself as a filter by eliminating from the list of likely abbreviations those strings that appear without a terminal period in the corpus.

The token-based and type-based heuristics combined produce an error rate of 1.65 %, which is still much higher than that of Punkt. An alternative approach evaluated by Grefenstette is to use a lexicon containing all ordinary words in the Brown corpus but no abbreviations or proper names in combination with the type-based heuristic for abbreviation detection. Even an approach with this massive amount of lexical knowledge still produces an error rate that is higher than that achieved by Punkt: 1.73 % versus 1.02 %. Only when he uses the complete lexicon and a list of common abbreviations, Grefenstette (1999) is able to attain a result that is slightly better than ours: His most resource intensive system achieves an error rate of 0.93 %.

These results indicate that Punkt can keep up well with rule-based systems even when tested on the specific language and text type that they have been developed for. Whereas rule-based systems either require extensive lexical resources or a large amount of manual labor, Punkt can be applied to new languages and corpora out of the box with no manual adaptation. Especially the fact that it can recognize new abbreviations on the fly is a great advantage because the results of the RE system on the Lacio-Web corpus and our own experiments in section 6.4.4 show that rule or list-based systems are often not sufficient to cover the productivity of abbreviation use in new corpora and languages; compare also Mikheev (2002, pages 298, 299, and 311) and Silla Jr. and Kaestner (2004).

7.2 Supervised Machine-Learning Systems

In this section, we discuss several supervised machine-learning approaches to sentence boundary detection described in the literature and compare their results to those achieved by Punkt. We regard those sentence boundary systems as supervised that require a set of manually disambiguated instances as training data.

7.2.1 Riley (1989) induces a decision tree for sentence boundary detection using the following features (pages 351 and 352):

- Prob[word with "." occurs at end of sentence]
- Prob[word after "." occurs at beginning of sentence]
- Length of word with "."
- Length of word after "."
- Case of word with ".": Upper, Lower, Cap, Numbers
- Case of word after ".": Upper, Lower, Cap, Numbers
- Punctuation after "." (if any)
- Abbreviation class of word with "."
  – e.g., month name, unit-of measure, title, address name, etc.

The resulting decision tree is able to classify the periods in the Brown corpus with a very low error rate of only 0.2 % which is 0.82 % better than that achieved by Punkt. However, the impressive performance of Riley's approach also requires impressive amounts of training data: He calculated the probabilities that a certain word occurs before or after a sentence boundary from 25 million words of AP newswire text. Such a large training corpus is probably not available for many languages; see also the comments in Palmer and Hearst (1997, page 245). Moreover, the last one of Riley's features, namely abbreviation class, requires quite specific lexical knowledge about abbreviation types which can only be taken from additional hand-crafted resources. It is unclear how well his approach would do with a realistic amount of training data and without these specific lexical resources.

7.2.2 The Satz system by Palmer and Hearst (1997) The Satz system[10] 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, which consist of documents with sentence boundaries annotated by hand, it can 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 were obtained using neural networks and decision trees.

We have used the same portion of the *Wall Street Journal* for evaluation in the preceding sections as Palmer and Hearst (1997) so that a direct comparison between the systems is possible. Palmer and Hearst report an error rate of 1.5 % for the initial version of their system, which uses neural nets as machine-learning algorithm, a lexicon of 30,000 words including an abbreviation list comprising 206 items, and was trained on a training set of 573 cases and a cross-validation set of 258 cases. This result is only slightly better than the error rate of 1.65 % which we obtained with Punkt. Their best result was produced by a version of their system that used decision tree induction, a lexicon of 5,000 words including the 206 abbreviations, and was trained on a set of 6,373 hand-annotated items. This configuration achieved an error rate of 1.0 % on the same test corpus. Palmer and Hearst have also evaluated their system on one French corpus and two German corpora. For the French corpus, they report an error rate of 0.4 %. On the two German corpora, their system produced error rates of 1.3 % and 0.5 %, respectively. Whereas the results of the Satz system for French are better than those for our system, their results on the two German corpora are worse than ours. However, as we were not able to use the same test corpora in our evaluation, these results are not directly comparable. Strunk, Silla Jr., and Kaestner (2006) give the results of a comparative evaluation of the present system against three other approaches, including Satz, on English and Brazilian Portuguese corpora. For both of these corpora, Satz achieved a slightly lower error rate than our system; compare Table 22.

While it is true that Satz performs somewhat better than Punkt in general, this is only the case if enough training data and additional resources are available. When it was not provided with a precompiled list of abbreviation, for example, it produced an error rate as high as 4.9 % on the WSJ corpus (Palmer and Hearst, 1997, page 255). This result combined with the results from section 6.4.4, which showed that general purpose abbreviation lists are not sufficient due to the productivity of abbreviation use, suggests that a reliable performance of the Satz system on new corpora can only be ensured if it is provided with an abbreviation list suitable for the domain in question and it is ideally trained on documents of the same genre as the corpus it will be tested on. Moreover, Palmer and Hearst report that Satz possesses a similar robustness with regard to single-case corpora as our system. However, this is again only the case if it has been retrained specifically on single-case corpora (Palmer and Hearst, 1997, pages 255, 256, 259).

7.2.3 MxTerminator Reynar and Ratnaparkhi (1997) use maximum-entropy modeling to learn contextual features from a hand-annotated training corpus that can be used to identify sentence boundaries. Their system, called MxTerminator,[11] employs features

10 Available from: http://elib.cs.berkeley.edu/src/satz/.
11 Available from: ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz.

Table 21
Comparison between MxTerminator (Reynar and Ratnaparkhi, 1997) and the Punkt System

Error <S>							
Corpus	**Cases**	**MxTerm.**	**Punkt**	**Corpus**	**Cases**	**MxTerm.**	**Punkt**
B. Portuegese	13,725	1.10 %	1.11 %	Italian	10,405	2.45 %	1.13 %
Dutch	18,068	1.13 %	0.97 %	Norwegian	25,531	1.34 %	0.81 %
English	24,282	1.53 %	1.65 %	Spanish	11,714	1.60 %	1.06 %
Estonian	23,243	2.79 %	2.12 %	Swedish	17,752	2.39 %	1.76 %
French	11,601	2.66 %	1.54 %	Turkish	18,942	1.77 %	1.31 %
German	34,256	0.63 %	0.35 %	**Mean Error**	MxTerm.: 1.76 %, Punkt: 1.26 %		

such as the token preceding a potential sentence boundary, the token following it, capitalization information about these tokens, whether one or both of them are abbreviations or not, and so on in its most portable version. It also induces a list of abbreviations from the training corpus by considering as an abbreviation every token in the training corpus that contains a possible end-of-sentence symbol but does not indicate a sentence boundary. This portable version does not depend on any lexical resources such as the part-of-speech information required by Satz. Reynar and Ratnaparkhi also built a version specialized for English newspaper text, which makes use of additional hand-crafted resources: a list of honorific abbreviations such as *Ms.* and *Dr.* and a list of corporate designating abbreviations such as *Corp.* and *S.p.A.* This specialized system achieved an error rate of 1.2 % on the English WSJ test corpus also used by Palmer and Hearst and by us. While this result is clearly better than that of our system, which produced an error rate of 1.65 %, MxTerminator had to be trained on 39,441 sentences of WSJ text and used hand-crafted lexical resources. For the more portable version of their system without language-specific abbreviation lists, Reynar and Ratnaparkhi report an error rate of 2.0 % on the same text corpus, an error rate, which is higher than that achieved by our system without training or lexical resources.

We have evaluated the portable version of MxTerminator on our eleven newspaper corpora using ten-fold cross-validation with nine tenths of the corpus as training set and one tenth as test set. Table 21 gives the average number of training cases and the mean error rates for MxTerminator on each corpus and compares them to those achieved by Punkt. Although MxTerminator achieves a slightly lower error rate on two corpora, namely Brazilian Portuguese and English, it produces an average error rate of 1.77 % on the newspaper corpora, which lies well above our system's mean error rate of 1.26 %, even though the number of training instances was sometimes as high as 34,256 (German) and never fell below 10,000. MxTerminator's performance was also worse than that of our system in a comparative evaluation on English and Brazilian Portuguese corpora described in Strunk, Silla Jr., and Kaestner (2006).

MxTerminator also shows that one cannot in general expect that the performance of a machine-learning system carries over to new corpora written in the same language without retraining. When Reynar and Ratnaparkhi evaluated their system on the Brown corpus without retraining, it achieved a relatively high error rate of 2.1 % (cf. Punkt's error rate of 1.02 %). The authors themselves remark:

> We present the Brown corpus performance to show the importance of training on the genre of text on which testing will be performed.

This points to the general problem that supervised systems that are not able to dynamically incorporate new knowledge, for example by discovering abbreviation types on the fly, cannot be expected to perform reliably on new corpora if the specific domain or genre is not known beforehand; compare also Mikheev (2002, pages 298, 314).

7.2.4 Stamatatos, Fakotakis, and Kokkinakis (1999) apply a specifically adapted version of transformation-based learning (Brill, 1995) to the problem of sentence boundary detection. For the initial-state annotation, they assume that every possible sentence-ending punctuation mark does indeed indicate a sentence boundary. In the first learning stage, their system uses characteristics such as identity, length, capitalization of the token containing a possible end-of-sentence boundary marker and of the token immediately following it as possible triggering environments to learn rules that remove sentence boundaries. In the second stage, it learns rules that reinsert sentence boundaries using the same space of possible triggering environments. Stamatatos, Fakotakis, and Kokkinakis (1999) trained their system on a hand-annotated corpus of Greek newspaper articles which contained 9,136 candidate punctuation marks. They tested it on a corpus containing articles from the same newspaper with 10,977 candidate punctuation marks and a lower bound of 79.6 %. Their system learned 312 rules in all and produced an error rate of 0.6 % on all punctuation marks, including periods, exclamation and question marks, and ellipses. When only periods and ellipses were considered, it achieved an error rate of 0.57 % with a set of 234 rules. Punkt achieved a respectable error rate of 1.50 % on this corpus without any training at all. As Stamatatos, Fakotakis, and Kokkinakis (1999) use individual words and brittle features such as capitalization information as triggering environments, their system is probably not very robust in that it requires training on a corpus that is very close to the texts that the system is intended to be used on in order to guarantee a low error rate.

7.3 Unsupervised Machine-Learning Systems

Unsupervised systems are approaches to sentence boundary detection that neither require specific hand-written rules or lexical resources nor have to be trained on hand-annotated training examples. Instead, they extract the required information for classification from the test corpus itself and/or additional unannotated text. We also regard our own approach as an unsupervised system.

7.3.1 Mikheev (2002) proposes to combine sentence boundary detection, proper name identification, and abbreviation detection in one system. He tackles the sentence boundary disambiguation task with a set of simple rules that can be applied after the tokens immediately to the left and to the right of a potential sentence boundary marker have been fully disambiguated. The most important questions are whether the token preceding a period is an abbreviation and whether the token following a period is a proper name. In order to answer these questions, he uses a combination of type-based abbreviation guessing heuristics, some of which have already been discussed in Grefenstette (1999), and what he calls the document-centered approach to abbreviation detection and proper name identification. This approach is based on the idea of classifying a candidate type as a whole as a proper name or an abbreviation based on instances of that type that occur in unambiguous contexts: For example, a type that always appears with a final period and occurs before a lower-cased word is likely to be an abbreviation, while a type that occurs with an uppercase first letter in the middle of a sentence is likely to be a proper name. He enhances these methods with the ability to distinguish between homographs (between an abbreviation and an ordinary word and between a proper name and an ordinary word) by collecting common sequences of more than one token that contain the candidate. His system requires some additional resources, namely a list of common words (i.e. not proper names), a list of common words that are frequent sentence starters, a list of frequent proper names that coincide with common words, and a domain-specific list of abbreviations, which can all be created from an unannotated corpus without human intervention.

Mikheev evaluated his system on the WSJ corpus and the Brown corpus of American English extracting the required additional resources from a 300,000 word corpus of New York Times articles. It achieved an error rate of 0.45 % on the WSJ corpus and an error rate of 0.28 % on the Brown corpus, while Punkt's error rates on these corpora were 1.65 % and 1.02 %, respectively.[12] Mikheev himself admits that the use of an additional domain-specific abbreviation list, which has to be recreated for every new domain, is not always possible, especially if the system is expected "to handle documents from unknown origin" (Mikheev, 2002, pages 305, 306). When his system was not equipped with an additional abbreviation list, the error rates rose to 1.41 % on the WSJ corpus and 0.65 % on the Brown corpus and were more comparable with those achieved by Punkt. Mikheev also tested his system in conjuction with a morphological analyzer on a corpus of BBC articles in Russian and obtained an error rate of 0.1 % for sentence boundary detection.

While Mikheev's system and Punkt are quite similar in spirit, his system uses advanced methods for the identification of proper names, while we have mostly concentrated on abbreviation detection. In fact, combining Mikheev's insights with our methods of abbreviation detection would most likely lead to another performance increase: Mikheev's abbreviation detection methods achieved an error rate of 6.6 % on the WSJ and an error rate of 8.9 % on the Brown corpus for the task of abbreviation detection when no additional abbreviation list was used. Even when such a list was consulted, his error rates – 0.8 % on the WSJ and 1.2 % on the Brown corpus – remained above those achieved by Punkt – 0.71 % and 0.82 %, respectively.

Abbreviation detection is also the area where we have spotted some critical issues in Mikheev's approach. Although Mikheev aims at a domain-independent system, he makes some decisions that could be harmful for domains other than news and literary fiction. He places an arbitrary length limit on possible abbreviations by applying his document-centered approach to abbreviation detection only to candidates that have a maximal length of four letters (Mikheev, 2002, page 299). This limit is already too strict for some abbreviations found in English newspaper corpora such as for example *Messrs., Calif., and Thurs.* The abbreviation *Calif.*(ornia), for instance, occurs 88 times in the portion of the WSJ corpus we used in our evaluation. The abbreviation lists that we have extracted from a German dictionary and a bilingual English-German dictionary (cf. section 6.4.4) show that such an arbitrary length limit can be quite problematic. In the German list, 28 % of all abbreviation types have a length greater than four. In the English list, 26 % are longer than four characters.

Moreover, Mikheev's document-centered approach is mostly based on capitalization information and therefore unable to uncover abbreviations that are always followed by a capitalized word, while our approach uses collocational information pertaining to the abbreviation itself and its final period and thus does not incur this problem. Mikheev's strong reliance on capitalization also renders his approach unsuitable for single-case text and leads to some problems for German where all nouns are always capitalized and not only proper names (Mikheev, 2002, page 315). Last but not least,

12 However, Mikheev seems to have tested on the whole WSJ portion of the Penn Treebank, while we have used only sections 03-06.

Table 22
Direct Comparison between Punkt and Other Systems for Sentence Boundary Detection

	F Measure		Error <S>	
System	**Tipster – WSJ**	**Lacio Web**	**Brown**	**WSJ**
Punkt	**91.51 %**	**97.22 %**	**1.02 %**	**1.65 %**
RE	91.78 %	89.87 %	–	–
Grefenstette & Tapanainen	–	–	0.93 %	–
Riley	–	–	0.20 %	–
Satz	91.88 %	99.16 %	–	1.00 %
MxTerminator	–	–	2.10 %	1.20 %
MxTerminator (portable)	91.22 %	96.46 %	2.50 %	2.00 %
Mikheev	–	–	0.28 %	0.45 %
Mikheev (tagger only)	–	–	0.98 %	1.95 %
Mikheev (+ tagger)	–	–	0.20 %	0.31 %

Mikheev's system does not include a specialized treatment of ordinal numbers, which we have shown to be quite important for some languages; compare section 6.4.5.

In Mikheev (2000) and Mikheev (2002), he describes the combination of a trigram tagger with his document-centered approach to abbreviation detection and proper name identification. After bootstrapping the training process on 20,000 words of tagged text, he trained the tagger in the unsupervised mode on the Brown corpus and evaluated it on the WSJ corpus and vice versa. When he used the tagger alone for the disambiguation of possible end-of-sentence marks, it achieved an error rate of 1.95 % on the WSJ corpus (vs. 1.65 % achieved by Punkt) and an error rate of 0.98 % on the Brown corpus (vs. 1.02 % achieved by our system). Enhancing the tagger with the heuristics to identify proper names and abbreviations described above improved the error rates to 0.31 % on the WSJ corpus and 0.20 % on the Brown corpus. When the heuristics were only applied to the test corpora themselves and no additional abbreviation list was employed, the resulting error rates were 1.39 % on the WSJ corpus and 0.65 % on the Brown corpus. When comparing these results achieved by using a part-of-speech tagger with those of our system it has to be kept in mind that although Mikheev trained his tagger in the unsupervised mode this technique normally still requires an extensive lexicon that contains the possible parts of speech for each lexical item, a resource, which is not always available for every language.

8 Conclusion

We have presented an unsupervised multilingual sentence boundary detection system that does not depend on any additional resources besides the corpus it is supposed to segment into sentences. It uses collocational information as a new type of evidence for the detection of abbreviations, initials, and ordinal numbers and is therefore much less dependent on orthographic information than competing systems. This fact enables it to accurately detect sentence boundaries even for single-case corpora. The experiments that we have carried out for eleven languages show that it is an accurate method well suited for different languages and different text genres. Although its performance is slightly inferior to the best results published in the literature on sentence boundary detection, compare Table 22, it can keep up well with rule-based methods and more straightforward supervised systems such as MxTerminator. Moreover, we were able to show that abbreviation use is quite productive and that systems that rely on resources such as abbreviation lists will have to be adapted specifically to every new language and every new domain, while no manual work and no language-specific resources are needed to adapt Punkt to new corpora. The error rate produced by Punkt on the WSJ corpus and the Brown corpus are higher than those produced by its direct competitor, the system described in Mikheev (2002). However, the fact that the two systems approach the problem differently and use complementary types of evidence suggests that combining our insights with Mikheev's might lead to an even better performance of unsupervised sentence boundary detection systems, especially since Punkt was better than Mikheev's system on the task of abbreviation detection.

Acknowledgments
We would like to thank Antti Arppe, Jørg Asmussen, Marti Hearst, Knut Hofland, Heiki-Jaan Kaalep, Cristina Mota, Umut Özge, Bilge Say, Efstathios Stamatatos, and particularly Katja Keßelmeier, Anneli von Könemann, and Kays Mutlu for their invaluable assistance in the construction of our test corpora. We are also grateful to three anonymous reviewers who gave us very helpful comments and suggestions.

References

Aluisio, Sandra M., Gisele M. Pinheiro, Marcelo Finger, Maria das Graças V. Nunes, and Stella E. O. Tagnin. 2003. The Lacio-Web Project: Overview and issues in Brazilian Portuguese corpora creation. In *Proceedings of Corpus Linguistics*, pages 14–21, Lancaster, UK.

Brill, Eric. 1995. Tranformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Dudenredaktion, editor. 2004. *Duden. Die deutsche Rechtschreibung (23rd ed.)*, volume 1 of *Der Duden in zwölf Bänden*. Dudenverlag, Mannheim.

Dunning, Ted. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Evert, Stefan and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 188–195, Toulouse, France.

Firth, John R. 1957. A synopsis of linguistic theory 1930-55. In *Studies in Linguistic Analysis (special volume of the Philological Society)*. The Philological Society, Oxford, pages 1–32.

Francis, W. Nelson and Henry Kucera. 1982. *Frequency Analysis of English Usage: Lexicon and Grammar*. Houghton Mifflin, New York.

Grefenstette, Gregory. 1999. Tokenization. In Hans van Halteren, editor, *Syntactic Wordclass Tagging*. Kluwer Academic Publishers, Dordrecht, pages 117–133.

Grefenstette, Gregory and Pasi Tapanainen. 1994. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, Budapest, Hungary.

Kiss, Tibor and Jan Strunk. 2002a. Scaled log likelihood ratios for the detection of abbreviations in text corpora. In *Proceedings of COLING 2002*, pages 1228–1232, Taipei, Taiwan.

Kiss, Tibor and Jan Strunk. 2002b. Viewing sentence boundary detection as collocation identification. In *Proceedings of KONVENS 2002*, pages 75–82, Saarbrücken, Germany.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

Mikheev, Andrei. 2000. Tagging sentence boundaries. In *Proceedings of ANLP-NAACL 2000*, pages 264–271, Seattle, Washington.

Mikheev, Andrei. 2002. Periods, capitalized words, etc. *Computational Linguistics*, 28(3):289–318.

Müller, Hans, V. Amerl, and G. Natalis. 1980. Worterkennungsverfahren als Grundlage einer Universalmethode zur automatischen Segmentierung von Texten in Sätze. Ein Verfahren zur maschinellen Satzgrenzenbestimmung im Englischen. *Sprache und Datenverarbeitung*, 4(1):46–64.

Nunberg, Geoffrey. 1990. *The Linguistics of Punctuation*, volume 18 of *CSLI Lecture Notes*. CSLI Publications, Stanford, California.

Palmer, David D. and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, 23(2):241–267.

Reynar, Jeffrey C. and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing*, pages 16–19, Washington, D.C.

Riley, Michael 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, Cape Cod, Massachusetts.

Silla Jr., Carlos N. and Celso A. A. Kaestner. 2004. An analysis of sentence boundary detection systems for English and Portuguese documents. In *Proceedings of CICLing 2004*, pages 135–141, Seoul, Korea.

Silla Jr., Carlos N., Jaime Dalla Valle Jr., and Celso A. A. Kaestner. 2003. Detecção automática de sentenças com o uso de expressões regulares. In *Proceedings of CBComp 2003*, pages 548–560, Itajaí, Brazil.

Stamatatos, Efstathios, Nikos Fakotakis, and George K. Kokkinakis. 1999. Automatic extraction of rules for sentence boundary disambiguation. In *Proceedings of the Workshop on Machine Learning in Human Language Technology*, pages 88–92, Chania, Greece.

Strunk, Jan, Carlos N. Silla Jr., and Celso A. A. Kaestner. 2006. A comparative evaluation of a new unsupervised sentence boundary detection approach on documents in English and Portuguese. In *Proceedings of CICLing 2006*, pages 132–143, Mexico City, Mexico.

van Rijsbergen, Cornelis J. 1979. *Information Retrieval*. Butterworths, London.

Walker, Daniel J., David E. Clements, Maki Darwin, and Jan W. Amtrup. 2001. Sentence boundary detection: A comparison of paradigms for improving MT quality. In *Proceedings of the MT Summit VIII*, Santiago de Compostela, Spain.

Willmann, Helmut and Heinz Messinger, editors. 1996. *Langenscheidts Großwörterbuch Englisch. Der Kleine Muret-Sanders (7th ed.)*, volume I: English–German. Langenscheidt, Berlin/Munich.