Periods, Capitalized Words, etc.

Andrei Mikheev*
University of Edinburgh

In this article we present an approach for tackling three important aspects of text normalization: sentence boundary disambiguation, disambiguation of capitalized words in positions where capitalization is expected, and identification of abbreviations. As opposed to the two dominant techniques of computing statistics or writing specialized grammars, our document-centered approach works by considering suggestive local contexts and repetitions of individual words within a document. This approach proved to be robust to domain shifts and new lexica and produced performance on the level with the highest reported results. When incorporated into a part-of-speech tagger, it helped reduce the error rate significantly on capitalized words and sentence boundaries. We also investigated the portability to other languages and obtained encouraging results.

1. Introduction

Disambiguation of sentence boundaries and normalization of capitalized words, as well as identification of abbreviations, however small in comparison to other tasks of text processing, are of primary importance in the developing of practical text-processing applications. These tasks are usually performed before actual "intelligent" text processing starts, and errors made at this stage are very likely to cause more errors at later stages and are therefore very dangerous.

Disambiguation of capitalized words in mixed-case texts has received little attention in the natural language processing and information retrieval communities, but in fact it plays an important role in many tasks. In mixed-case texts capitalized words usually denote proper names (names of organizations, locations, people, artifacts, etc.), but there are special positions in the text where capitalization is expected. Such mandatory positions include the first word in a sentence, words in titles with all significant words capitalized or table entries, a capitalized word after a colon or open quote, and the first word in a list entry, among others. Capitalized words in these and some other positions present a case of ambiguity: they can stand for proper names, as in *White later said* ..., or they can be just capitalized common words, as in *White elephants are* .... The **disambiguation of capitalized words** in ambiguous positions leads to the **identification of proper names** (or their derivatives), and in this article we will use these two terms and the term **case normalization** interchangeably.

Church (1995, p. 294) studied, among other simple text normalization techniques, the effect of case normalization for different words and showed that "sometimes case variants refer to the same thing (*hurricane* and *Hurricane*), sometimes they refer to different things (*continental* and *Continental*) and sometimes they don't refer to much of anything (e.g., *anytime* and *Anytime*)." Obviously these differences arise because some capitalized words stand for proper names (such as *Continental*, the name of an airline) and some do not.

* Institute for Communicating and Collaborative Systems, Division of Informatics, 2 Buccleuch Place, Edinburgh EH8 9LW, UK. E-mail: mikheev@cogsci.ed.ac.uk

© 2002 Association for Computational Linguistics

Proper names are the main concern of the named-entity recognition subtask (Chinchor 1998) of information extraction. The main objective of this subtask is the identification of proper names and also their classification into semantic categories (person, organization, location, etc.).[1] There the disambiguation of the first word in a sentence (and in other ambiguous positions) is one of the central problems: about 20% of named entities occur in ambiguous positions. For instance, the word *Black* in the sentence-initial position can stand for a person's surname but can also refer to the color. Even in multiword capitalized phrases, the first word can belong to the rest of the phrase or can be just an external modifier. In the sentence *Daily, Mason and Partners lost their court case*, it is clear that *Daily, Mason and Partners* is the name of a company. In the sentence *Unfortunately, Mason and Partners lost their court case*, the name of the company does not include the word *Unfortunately*, but the word *Daily* is just as common a word as *Unfortunately*.

Identification of proper names is also important in machine translation, because usually proper names are transliterated (i.e., phonetically translated) rather than properly (semantically) translated. In confidential texts, such as medical records, proper names must be identified and removed before making such texts available to people unauthorized to have access to personally identifiable information. And in general, most tasks that involve text analysis will benefit from the robust disambiguation of capitalized words into proper names and common words.

Another important task of text normalization is sentence boundary disambiguation (SBD) or sentence splitting. Segmenting text into sentences is an important aspect in developing many applications: syntactic parsing, information extraction, machine translation, question answering, text alignment, document summarization, etc. Sentence splitting in most cases is a simple matter: a period, an exclamation mark, or a question mark usually signals a sentence boundary. In certain cases, however, a period denotes a decimal point or is a part of an abbreviation, and thus it does not necessarily signal a sentence boundary. Furthermore, an abbreviation itself can be the last token in a sentence in which case its period acts at the same time as part of this abbreviation and as the end-of-sentence indicator (fullstop). A detailed introduction to the SBD problem can be found in Palmer and Hearst (1997).

The disambiguation of capitalized words and sentence boundaries presents a chicken-and-egg problem. If we know that a capitalized word that follows a period is a common word, we can safely assign such period as sentence terminal. On the other hand, if we know that a period is not sentence terminal, then we can conclude that the following capitalized word is a proper name.

Another frequent source of ambiguity in end-of-sentence marking is introduced by abbreviations: if we know that the word that precedes a period is *not* an abbreviation, then almost certainly this period denotes a sentence boundary. If, however, this word is an abbreviation, then it is not that easy to make a clear decision. This problem is exacerbated by the fact that abbreviations do not form a closed set; that is, one cannot list all possible abbreviations. Moreover, abbreviations can coincide with regular words; for example, "in" can denote an abbreviation for "inches," "no" can denote an abbreviation for "number," and "bus" can denote an abbreviation for "business."

In this article we present a method that tackles sentence boundaries, capitalized words, and abbreviations in a uniform way through a document-centered approach. As opposed to the two dominant techniques of computing statistics about the words that surround potential sentence boundaries or writing specialized grammars, our ap-

1 In this article we are concerned only with the identification of proper names.

proach disambiguates capitalized words and abbreviations by considering suggestive local contexts and repetitions of individual words within a document. It then applies this information to identify sentence boundaries using a small set of rules.

2. Performance Measure, Corpora for Evaluation, and Intended Markup

A standard practice for measuring the performance of a system for the class of tasks with which we are concerned in this article is to calculate its **error rate**:

$$error\_rate = \frac{incorrectly\_assigned}{all\_assigned\_by\_system}$$

This single measure gives enough information, provided that the system does not leave unassigned word tokens that it is intended to handle. Obviously, we want the system to handle all cases as accurately as possible. Sometimes, however, it is beneficial to assign only cases in which the system is confident enough, leaving the rest to be handled by other methods. In this case apart from the error rate (which corresponds to precision or accuracy as 1−error rate) we also measure the system's **coverage** or **recall**

$$coverage = \frac{correctly\_assigned}{all\_to\_be\_assigned}$$

2.1 Corpora for Evaluation

There are two corpora normally used for evaluation in a number of text-processing tasks: the Brown corpus (Francis and Kucera 1982) and the *Wall Street Journal* (WSJ) corpus, both part of the Penn Treebank (Marcus, Marcinkiewicz, and Santorini 1993). The Brown corpus represents general English. It contains over one million word tokens and is composed from 15 subcorpora that belong to different genres and domains, ranging from news wire texts and scientific papers to fiction and transcribed speech. The Brown corpus is rich in out-of-vocabulary (unknown) words, spelling errors, and ungrammatical sentences with complex internal structure. Altogether there are about 500 documents in the Brown corpus, with an average length of 2,300 word tokens.

The WSJ corpus represents journalistic news wire style. Its size is also over a million word tokens, and the documents it contains are rich in abbreviations and proper names, but they are much shorter than those in the Brown corpus. Altogether there are about 2,500 documents in the WSJ corpus, with an average length of about 500 word tokens.

Documents in the Penn Treebank are segmented into paragraphs and sentences. Sentences are further segmented into word tokens annotated with part-of-speech (POS) information. POS information can be used to distinguish between proper names and common words. We considered proper nouns (`NNP`), plural proper nouns (`NNPS`), and proper adjectives[2] (`JJP`) to signal proper names, and all other categories were considered to signal common words or punctuation. Since proper adjectives are not included in the Penn Treebank tag set, we had to identify and retag them ourselves with the help of a gazetteer.

Abbreviations in the Penn Treebank are tokenized together with their trailing periods, whereas fullstops and other sentence boundary punctuation are tokenized as separate tokens. This gives all necessary information for the evaluation in all our three

2 These are adjectives derived from proper nouns (e.g. "American").

tasks: the sentence boundary disambiguation task, the capitalized word disambiguation task, and the abbreviation identification task.

2.2 Tokenization Convention and Corpora Markup

For easier handling of potential sentence boundary punctuation, we developed a new tokenization convention for periods. In the traditional Penn Treebank schema, abbreviations are tokenized *together* with their trailing periods, and thus stand-alone periods unambiguously signal the end of a sentence. We decided to treat periods and all other potential sentence termination punctuation as "first-class citizens" and adopted a convention always to tokenize a period (and other punctuation) as a separate token when it is followed by a white space, line break, or punctuation. In the original Penn Treebank format, periods are unambiguous, whereas in our new convention they can take on one of the three tags: fullstop (`.`), part of abbreviation (`A`) or both (`*`).

To generate the new format from the Penn Treebank, we had to split final periods from abbreviations, mark them as separate tokens and assign them with `A` or `*` tags according to whether or not the abbreviation was the last token in a sentence. We applied a similar tokenization convention to the case in which several (usually three) periods signal ellipsis in a sentence. Again, sometimes such constructions occur within a sentence and sometimes at a sentence break. We decided to treat such constructions similarly to abbreviations, tokenize all periods but the last together in a single token, and tokenize the last period separately and tag it with `A` or `*` according to whether or not the ellipsis was the last token in a sentence. We treated periods in numbers (e.g., *14.534*) or inside acronyms (e.g., *Y.M.C.A.*) as part of tokens rather than separate periods.

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

We also specially marked word tokens in positions where they were ambiguously capitalized if such word tokens occurred in one of the following contexts:

- the first token in a sentence
- following a separately tokenized period, question mark, exclamation mark, semicolon, colon, opening quote, closing quote, opening bracket, or closed bracket
- occurring in a sentence with all words capitalized

All described transformations were performed automatically by applying a simple `Perl` script. We found quite a few infelicities in the original tokenization and tagging, however, which we had to correct by hand. We also converted both our corpora from their original Penn Treebank format into an XML format where each word token is represented as an XML element (`W`) with the attribute `C` holding its POS information and attribute `A` set to `Y` for ambiguously capitalized words. An example of such a markup is displayed in Figure 1.

3. Our Approach to Sentence Boundary Disambiguation

If we had at our disposal entirely correct information on whether or not each word preceding a period was an abbreviation and whether or not each capitalized word

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

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

...<W C=',',</W> <W C=NNP>Tex</W><W C='*'>.</W>
   <W A=Y C=JJP>American</W>...
```

Figure 1
Example of the new tokenization and markup generated from the Penn Treebank format. Tokens are represented as XML elements `W`, where the attribute `C` holds POS information. Proper names are tagged as `NNP`, `NNPS` and `JJP`. Periods are tagged as `.` (fullstop), `A` (part of abbreviation), `*` (a fullstop and part of abbreviation at the same time). Ambiguously capitalized words are marked with `A = Y`.

that follows a period was a proper name, we could apply a very simple set of rules to disambiguate sentence boundaries:

- If a period follows a nonabbreviation, it is sentence terminal (`.`).
- If a period follows an abbreviation and is the last token in a text passage (paragraph, document, etc.), it is sentence terminal and part of the abbreviation (`*`).
- If a period follows an abbreviation and is not followed by a capitalized word, it is part of the abbreviation and is not sentence terminal (`A`).
- If a period follows an abbreviation and is followed by a capitalized word that is not a proper name, it is sentence terminal and part of the abbreviation (`*`).

It is a trivial matter to extend these rules to allow for brackets and quotation marks between the period and the following word. To handle other sentence termination punctuation such as question and exclamation marks and semicolons, this rule set also needs to include corresponding rules. The entire rule set for sentence boundary disambiguation that was used in our experiments is listed in Appendix A.

3.1 Ideal Case: Upper Bound for Our SBD Approach

The estimates from the Brown corpus and the WSJ corpus (section 3) show that the application of the SBD rule set described above together with the information on abbreviations and proper names marked up in the corpora produces very accurate results (error rate less than 0.0001%), but it leaves unassigned the outcome of the case in which an abbreviation is followed by a proper name. This is a truly ambiguous case, and to deal with this situation in general, one should encode detailed information about the words participating in such contexts. For instance, honorific abbreviations such as *Mr.* or *Dr.* when followed by a proper name almost certainly do not end a sentence, whereas the abbreviations of U.S. states such as *Mo.*, *Cal.*, and *Ore.*, when followed directly by a proper name, most likely end a sentence. Obviously encoding this kind of information into the system requires detailed analysis of the domain lexica, is not robust to unseen abbreviations, and is labor intensive.

To make our method robust to unseen words, we opted for a crude but simple solution. If such ambiguous cases are always resolved as "not sentence boundary" (`A`), this produces, by our measure, an error rate of less than 3%. Estimates from the Brown

Table 1
Estimates of the upper and lower bound error rates on the SBD task for our method. Three estimated categories are sentence boundaries, ambiguously capitalized words, and abbreviations.

		Brown Corpus			WSJ Corpus		
		SBD	Amb. Cap.	Abbreviations	SBD	Amb. Cap.	Abbreviations
	Number of resolved instances	59,539	58,957	4,657	53,043	54,537	16,317
A	Upper Bound: All correct proper names All correct abbrs.	**0.01%**	0.0%	0.0%	0.13%	0.0%	0.0%
B	Lower Bound: Lookup proper names Guessed abbrs.	**2.00%**	7.4%	10.8%	4.10%	15.0%	9.6%
C	Lookup proper names All correct abbrs.	1.20%	7.4%	0.0%	2.34%	15.0%	0.0%
D	All correct proper names Guessed abbrs.	0.45%	0.0%	10.8%	1.96%	0.0%	9.6%

corpus and the WSJ corpus showed that such ambiguous cases constitute only 5–7% of all potential sentence boundaries. This translates into a relatively small impact of the crude strategy on the overall error rate on sentence boundaries. This impact was measured at 0.01% on the Brown corpus and at 0.13% on the WSJ corpus, as presented in row A of Table 1. Although this overly simplistic strategy extracts a small penalty from the performance, we decided to use it because it is very general and independent of domain-specific knowledge.

The SBD handling strategy described above is simple, robust, and well performing, but it relies on the assumption that we have entirely correct information about abbreviations and proper names, as can be seen in row A of the table. The main difficulty is that when dealing with real-world texts, we have to identify abbreviations and proper names ourselves. Thus estimates based on the application of our method when using 100% correctly disambiguated capitalized words and abbreviations can be considered as the upper bound for the SBD approach, that is, the top performance we can achieve.

3.2 Worst Case: Lower Bound for Our SBD Approach

We can also estimate the lower bound for this approach applying very simple strategies to the identification of proper names and abbreviations.

The simplest strategy for deciding whether or not a capitalized word in an ambiguous position is a proper name is to apply a lexical-lookup strategy (possibly enhanced with a morphological word guesser, e.g., Mikheev [1997]). Using this strategy, words not listed as known common words for a language are usually marked as proper names. The application of this strategy produced a 7.4% error rate on the Brown corpus and a 15% error rate on the WSJ corpus. The difference in error rates can be explained by the observation that the WSJ corpus contains a higher percentage of organization names and person names, which often coincide with common English words,

and it contains more words in titles with all important words capitalized, which we also consider as ambiguously capitalized.

The simplest strategy for deciding whether a word that is followed by a period is an abbreviation or a regular word is to apply well-known heuristics based on the observation that single-word abbreviations are short and normally do not include vowels (*Mr.*, *Dr.*, *kg.*). Thus a word without vowels can be guessed to be an abbreviation unless it is written in all capital letters and can stand for an acronym or a proper name (e.g., *BBC*). A span of single letters separated by periods forms an abbreviation too (e.g., *Y.M.C.A.*). A single letter followed by a period is also a very likely abbreviation. There is also an additional heuristic that classifies as abbreviations short words (with length less than five characters) that are followed by a period and then by a comma, a lower-cased word, or a number. All other words are considered to be nonabbreviations.

These heuristics are reasonably accurate. On the WSJ corpus they misrecognized as abbreviations only 0.2% of tokens. On the Brown corpus the misrecognition rate was significantly higher: 1.6%. The major source for these errors were single letters that stand for mathematical symbols in the scientific subcorpora of the Brown Corpus (e.g., *point T* or *triangle F*). The major shortcoming of these abbreviation-guessing heuristics, however, comes from the fact that they failed to identify about 9.5% of abbreviations. This brings the overall error rate of the abbreviation-guessing heuristics to about 10%.

Combining the information produced by the lexical-lookup approach to proper name identification with the abbreviation-guessing heuristics feeding the SBD rule set gave us a 2.0% error rate on the Brown corpus and 4.1% on the WSJ corpus on the SBD task. This can be interpreted as the lower bound to our SBD approach. Here we see how errors in the identification of proper names and abbreviations propagated themselves into errors on sentence boundaries. Row B of Table 1 displays a summary for the lower-bound results.

3.3 Major Findings

We also measured the importance of each of the two knowledge sources (abbreviations and proper names) separately. First, we applied the SBD rule set when all abbreviations were correctly identified (using the information presented in the corpus) but applying the lexical lookup strategy to proper-name identification (row C of Table 1). Then, we applied the SBD rule set when all proper names were correctly identified (using the information presented in the corpus) but applying the guessing heuristics to handle abbreviations (row D of the table). In general, when a knowledge source returned 100% accurate information this significantly improved performance on the SBD task measured against the lower-bound error rate. We also see that proper names have a higher impact on the SBD task than abbreviations.

Since the upper bound of our SBD approach is high and the lower bound is far from being acceptable, *our main strategy for sentence boundary disambiguation will be to invest in the disambiguation of capitalized words and abbreviations that then feed our SBD rule set*.

4. Document-Centered Approach to Proper Name and Abbreviation Handling

As we discussed above, virtually any common word can potentially act as a proper name or part of a multiword proper name. The same applies to abbreviations: there is no fixed list of abbreviations, and almost any short word can be used as an abbreviation. Fortunately, there is a mitigating factor too: important words are typically used in a document more than once and in different contexts. Some of these contexts create

ambiguity, but some do not. Furthermore, ambiguous words and phrases are usually unambiguously introduced at least once in the text unless they are part of common knowledge presupposed to be possessed by the readers.

This observation can be applied to a broader class of tasks. For example, people are often referred to by their surnames (e.g., *Black*) but are usually introduced at least once in the text either with their first name (*John Black*) or with their title/profession affiliation (*Mr. Black*, *President Bush*), and it is only when their names are common knowledge that they do not need an introduction (e.g., *Castro*, *Gorbachev*). Thus our suggestion is to *look at the unambiguous usages of the words in question in the entire document*.

In the case of proper name identification, we are not concerned with the semantic class of a name (e.g., whether it is a person's name or a location), but rather we simply want to distinguish whether a capitalized word in a particular occurrence acts as a proper name (or part of a multiword proper name). If we restrict our scope to a single sentence, we might find that there is just not enough information to make a reliable decision. For instance, *Riders* in the sentence *Riders rode all over the green* is equally likely to be a proper noun, a plural proper noun, or a plural common noun. But if in the same text we find *John Riders*, this sharply increases the likelihood that the proper noun interpretation is the correct one, and conversely if we find *many riders*, this suggests the plural-noun interpretation.

The above reasoning can be summarized as follows: if we detect that a word is used capitalized in an unambiguous context, this increases the chances that this word acts as a proper name in ambiguous positions in the same document. And conversely if a word is seen only lower-cased, this increases the chances that it should be treated as a common word even when used capitalized in ambiguous positions in the same document. (This, of course, is only a general principle and will be further elaborated elsewhere in the article.)

The same logic applies to abbreviations. Although a short word followed by a period is a potential abbreviation, the same word occurring in the same document in a different context can be unambiguously classified as a regular word if it is used without a trailing period, or it can be unambiguously classified as an abbreviation if it is used with a trailing period and is followed by a lower-cased word or a comma. This information gives us a better chance of assigning these potential abbreviations correctly in nonobvious contexts.

We call such style of processing a **document-centered approach** (DCA), since information for the disambiguation of an individual word token is derived from the entire document rather than from its immediate local context. Essentially the system collects suggestive instances of usage for target words from each document under processing and applies this information on the fly to the processing of the document, in a manner similar to instance-based learning. This differentiates DCA from the traditional corpus-based approach, in which learning is applied prior to processing, which is usually performed with supervision over multiple documents of the training corpus.

5. Building Support Resources

Our method requires only four word lists. Each list is a collection of words that belong to a single type, but at the same time, a word can belong to multiple lists. Since we have four lists, we have four types:

- common word (as opposed to proper name)
- common word that is a frequent sentence starter

- frequent proper name
- abbreviation (as opposed to regular word)

These four lists can be acquired completely automatically from raw (unlabeled) texts. For the development of these lists we used a collection of texts of about 300,000 words derived from the *New York Times* (NYT) corpus that was supplied as training data for the 7th Message Understanding Conference (MUC-7) (Chinchor 1998). We used these texts because the approach described in this article was initially designed to be part of a named-entity recognition system (Mikheev, Grover, and Moens 1998) developed for MUC-7. Although the corpus size of 300,000 words can be seen as large, the fact that this corpus does not have to be annotated in any way and that a corpus of similar size can be easily collected from on-line sources (including the Internet) makes this resource cheap to obtain.

The first list on which our method relies is a **list of common words**. This list includes common words for a given language, but no supplementary information such as POS or morphological information is required to be present in this list. A variety of such lists for many languages are already available (e.g., Burnage 1990). Words in such lists are usually supplemented with morphological and POS information (which is not required by our method). We do not have to rely on pre-existing resources, however. A list of common words can be easily obtained automatically from a raw (unannotated in any way) text collection by simply collecting and counting lower-cased words in it. We generated such list from the NYT collection. To account for potential spelling and capitalization errors, we included in the list of common words only words that occurred lower-cased at least three times in the NYT texts. The list of common words that we developed from the NYT collection contained about 15,000 English words.

The second list on which our method relies is a **frequent-starters list**, a list of common words most frequently used in sentence-starting positions. This list can also be obtained completely automatically from an unannotated corpus by applying the lexical-lookup strategy. As discussed in Section 3.2, this strategy performs with a 7–15% error rate. We applied the list of common words over the NYT text collection to tag capitalized words in sentence-starting positions as common words and as proper names: if a capitalized word was found in the list of common words, it was tagged as a common word: otherwise it was tagged as a proper name. Of course, such tagging was far from perfect, but it was good enough for our purposes. We included in the frequent-starters list only the 200 most frequent sentence-starting common words. This was more than a safe threshold to ensure that no wrongly tagged words were added to this list. As one might predict, the most frequent sentence-starting common word was *The*. This list also included some adverbs, such as *However*, *Suddenly*, and *Once*; some prepositions, such as *In*, *To*, and *By*; and even a few verbs: *Let*, *Have*, *Do*, etc.

The third list on which our method relies is a **list of single-word proper names** that coincide with common words. For instance, the word *Japan* is much more likely to be used as a proper name (name of a country) rather than a verb, and therefore it needs to be included in this list. We included in the proper name list 200 words that were most frequently seen in the NYT text collection as single capitalized words in unambiguous positions and that at the same time were present in the list of common words. For instance, the word *The* can frequently be seen capitalized in unambiguous positions, but it is always followed by another capitalized word, so we do not count it as a candidate. On the other hand the word *China* is often seen capitalized in unambiguous positions where it is not preceded or followed by other capitalized words. Since *china*

Table 2
Error rates for different combinations of the abbreviation identification methods, including combinations of guessing heuristics (GH), lexical lookup (LL), and the document-centered approach (DCA).

	Abbreviation Identification Method	WSJ	Brown
A	GH	9.6%	10.8%
B	LL	12.6%	11.9%
C	GH + LL	1.2%	2.1%
D	GH + DCA	6.6%	8.9%
E	GH + DCA + LL	0.8%	1.2%

is also listed among common words and is much less frequently used in this way, we include it in the proper name list.

The fourth list on which our method relies is a **list of known abbreviations**. Again, we induced this list completely automatically from an unannotated corpus. We applied the abbreviation-guessing heuristics described in Section 6 to our NYT text collection and then extracted the 270 most frequent abbreviations: all abbreviations that appeared five times or more. This list included honorific abbreviations (*Mr*, *Dr*), corporate designators (*Ltd*, *Co*), month name abbreviations (*Jan*, *Feb*), abbreviations of names of U.S. states (*Ala*, *Cal*), measure unit abbreviations (*ft*, *kg*), etc. Although we described these abbreviations in groups, this information was not encoded in the list; the only information this list provides is that a word is a known abbreviation.

Among these four lists the first three reflect general language regularities and usually do not require modification for handling texts from a new domain. The abbreviation list, however, is much more domain dependent and for better performance needs to be reinduced for a new domain. Since the compilation of all four lists does not require data preannotated in any way, it is very easy to specialize the above-described lists to a particular domain: we can simply rebuild the lists using a domain-specific corpus. This process is completely automatic and does not require any human labor apart from collecting a raw domain-specific corpus. Since all cutoff thresholds that we applied here were chosen by intuition, however, different domains might require some new settings.

6. Recognizing Abbreviations

The answer to the question of whether or not a particular word token is an abbreviation or a regular word largely solves the sentence boundary problem. In the Brown corpus 92% of potential sentence boundaries come after regular words. The WSJ corpus is richer with abbreviations, and only 83% of sentences in that corpus end with a regular word followed by a period. In Section 3 we described the heuristics for abbreviation guessing and pointed out that although these heuristics are reasonably accurate, they fail to identify about 9.5% of abbreviations. Since unidentified abbreviations are then treated as regular words, the overall error rate of the guessing heuristics was measured at about 10% (row A of Table 2). Thus, to improve this error rate, we need first of all to improve the coverage of the abbreviation-handling strategy.

A standard way to do this is to use the guessing heuristics in conjunction with a list of known abbreviations. We decided to use the list of 270 abbreviations described in Section 5. First we applied only the lexical-lookup strategy to our two corpora (i.e.,

only when a token was found in the list of 270 known abbreviations was it marked as an abbreviation). This gave us an unexpectedly high error rate of about 12%, as displayed in row B of Table 2. When we investigated the reason for the high error rate, we found that the majority of single letters and spans of single letters separated by periods (e.g. Y.M.C.A.) found in the Brown corpus and the WSJ corpus were not present in our abbreviation list and therefore were not recognized as abbreviations.

Such cases, however, are handled well by the abbreviation-guessing heuristics. When we applied the abbreviation list together with the abbreviation-guessing heuristics (row C of Table 2), this gave us a very strong performance on the WSJ corpus: an error rate of 1.2%. On the Brown corpus, the error rate was higher: 2.1%. This can be explained by the fact that we collected our abbreviation list from a corpus of news articles that is not too dissimilar to the texts in the WSJ corpus and thus, this list contained many abbreviations found in that corpus. The Brown corpus, in contrast, ranges across several different domains and sublanguages, which makes it more difficult to compile a list from a single corpus to cover it.

6.1 Unigram DCA

The abbreviation-guessing heuristics supplemented with a list of abbreviations are accurate, but they still can miss some abbreviations. For instance, if an abbreviation like *sec* or *Okla.* is followed by a capitalized word and is not listed in the list of abbreviations, the guessing heuristics will not uncover them. We also would like to boost the abbreviation handling with a domain-independent method that enables the system to function even when the abbreviation list is not much of a help. Thus, in addition to the list of known abbreviations and the guessing heuristics, we decided to apply the DCA as described below.

Each word of length four characters or less that is followed by a period is treated as a potential abbreviation. First, the system collects unigrams of potential abbreviations in unambiguous contexts from the document under processing. If a potential abbreviation is used elsewhere in the document without a trailing period, we can conclude that it in fact is not an abbreviation but rather a regular word (nonabbreviation). To decide whether a potential abbreviation is really an abbreviation, we look for contexts in which it is followed by a period and then by a lower-cased word, a number, or a comma.

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

6.2 Bigram DCA

The DCA relies on the assumption that there is a consistency in writing within the same document. Different authors can write *Mr* or *Dr* with or without a trailing period, but we assume that the same author (the author of a particular document) writes it in the same way consistently. A situation can arise, however, in which the same potential abbreviation is used as a regular word and as an abbreviation within the same

document. This is usually the case when an abbreviation coincides with a regular word, for example *Sun.* (meaning Sunday) and *Sun* (the name of a newspaper). To tackle this problem, the system can collect from a document not only unigrams of potential abbreviations in unambiguous contexts but also their bigrams with the preceding word. Of course, as in the case with unigrams, the bigrams are collected on the fly and completely automatically.

For instance, if the system finds a context *vitamin C is*, it stores the bigram *vitamin C* and the unigram *C* with the information that *C* is a regular word. If in the same document the system also detects a context *John C. later said*, it stores the bigram *John C* and the unigram *C* with the information that *C* is an abbreviation. Here we have conflicting information for the word *C*: it was detected to act as a regular word and as an abbreviation within the same document, so there is not enough information to resolve ambiguous cases purely using the unigram. Some cases, however, can still be resolved on the basis of the bigrams. The system will assign *C* as a regular word (nonabbreviation) in an ambiguous context such as *vitamin C. Research* because of the stored *vitamin C* bigram. Obviously from such a short context, it is difficult even for a human to make a confident decision, but the evidence gathered from the entire document can influence this decision with a high degree of confidence.

6.3 Resulting Approach

When neither unigrams nor bigrams can help to resolve an ambiguous context for a potential abbreviation, the system decides in favor of the more frequent category for that abbreviation. If the word *In* was detected to act as a regular word (preposition) five times in the current document and two times as abbreviation (for the state *Indiana*), in a context in which neither of the bigrams collected from the document can be applied, *In* is assigned as a regular word (nonabbreviation). The last-resort strategy is to assign all nonresolved cases as nonabbreviations.

Row D of Table 2 shows the results when we applied the abbreviation-guessing heuristics together with the DCA. On the WSJ corpus, the DCA reduced the error rate of the guessing heuristics alone (row A) by about 30%; on the Brown corpus its impact was somewhat smaller, about 18%. This can be explained by the fact that abbreviations in the WSJ corpus have a much higher repetition rate, which is very important for the DCA.

We also applied the DCA together with the lexical lookup and the guessing heuristics. This reduced the error rate on abbreviation identification by about 30% in comparison with the list and guessing heuristics configuration, as can be seen in row E of Table 2.

7. Disambiguating Capitalized Words

The second key task of our approach is the disambiguation of capitalized words that follow a potential sentence boundary punctuation sign. Apart from being an important component in the task of text normalization, information about whether or not a capitalized word that follows a period is a common word is crucial for the SBD task, as we showed in Section 3. We tackle capitalized words in a similar fashion as we tackled the abbreviations: through a document-centered approach that analyzes on the fly the distribution of ambiguously capitalized words in the entire document. This is implemented as a cascade of simple strategies, which were briefly described in Mikheev (1999).

7.1 The Sequence Strategy

The first DCA strategy for the disambiguation of ambiguous capitalized words is to explore sequences of words extracted from contexts in which the same words are used unambiguously with respect to their capitalization. We call this the **sequence strategy**. The rationale behind this strategy is that if there is a phrase of two or more capitalized words starting from an unambiguous position (e.g., following a lower-cased word), the system can be reasonably confident that even when the same phrase starts from an unreliable position (e.g., after a period), all its words still have to be grouped together and hence are proper nouns. Moreover, this applies not just to the exact replication of the capitalized phrase, but to any partial ordering of its words of size two characters or more preserving their sequence.

For instance, if a phrase *Rocket Systems Development Co.* is found in a document starting from an unambiguous position (e.g., after a lower-cased word, a number, or a comma), the system collects it and also generates its partial-order subphrases: *Rocket Systems*, *Rocket Systems Co.*, *Rocket Co.*, *Systems Development*, etc. If then in the same document *Rocket Systems* is found in an ambiguous position (e.g., after a period), the system will assign the word *Rocket* as a proper noun because it is part of a multiword proper name that was seen in the unambiguous context.

A span of capitalized words can also internally include alpha-numerals, abbreviations with internal periods, symbols, and lower-cased words of length three characters or shorter. This enables the system to capture phrases like *A & M* and *The Phantom of the Opera*. Partial orders from such phrases are generated in a similar way, but with the restriction that every generated subphrase should start and end with a capitalized word.

The sequence strategy can also be applied to disambiguate common words. Since in the case of common words the system cannot determine boundaries of a phrase, only bigrams of the lower-cased words with their following words are collected from the document. For instance, from a context *continental suppliers of Mercury*, the system collects three bigrams: *continental suppliers*, *suppliers of*, and *of Mercury*. When the system encounters the phrase *Continental suppliers* after a period, it can now use the information that in the previously stored bigram *continental suppliers*, the word token *continental* was written lower-cased and therefore was unambiguously used as a common word. On this basis the system can assign the ambiguous capitalized word token *Continental* as a common word.

Row A of Table 3 displays the results obtained in the application of the sequence strategy to the Brown corpus and the WSJ corpus. The sequence strategy is extremely useful when we are dealing with names of organizations, since many of them are multiword phrases composed of common words. For instance, the words *Rocket* and *Insurance* can be used both as proper names and common words within the same document. The sequence strategy maintains contexts of the usages of such words within the same document, and thus it can disambiguate such usages in the ambiguous positions matching surrounding words. And indeed, the error rate of this strategy when applied to proper names was below 1%, with coverage of about 9–12%.

For tagging common words the sequence strategy was also very accurate (error rate less than 0.3%), covering 17% of ambiguous capitalized common words on the WSJ corpus and 25% on the Brown corpus. The higher coverage on the Brown corpus can be explained by the fact that the documents in that corpus are in general longer than those in the WSJ corpus, which enables more word bigrams to be collected from a document.

Dual application of the sequence strategy contributes to its robustness against potential capitalization errors in the document. The negative evidence (not proper name)

Table 3
First part: Error rates of different individual strategies for capitalized-word disambiguation. Second part: Error rates of the overall cascading application of the individual strategies.

	Strategy	Word Class	Error Rate		Coverage	
			WSJ	Brown	WSJ	Brown
A	Sequence strategy	Proper Names	0.12%	0.97%	12.6%	8.82%
	Sequence strategy	Common Words	0.28%	0.21%	17.68%	26.5%
B	Frequent-list lookup strategy	Proper Names	0.49%	0.16%	2.62%	6.54%
	Frequent-list lookup strategy	Common Words	0.21%	0.14%	64.62%	61.20%
C	Single-word assignment strategy	Proper Names	3.18%	1.96%	18.77%	34.13%
	Single-word assignment strategy	Common Words	6.51%	2.87%	3.07%	4.78%
D	Cascading DCA	Proper/Common	1.10%	0.76%	84.12%	91.76%
E	Cascading DCA and lexical lookup	Proper/Common	4.88%	2.83%	100.0%	100.0%

is used together with the positive evidence (proper name) and blocks assignment when conflicts are found. For instance, if the system detects a capitalized phrase *The President* in an unambiguous position, then the sequence strategy will treat the word *the* as part of the proper name *The President* even when this phrase follows a period. If in the same document, however, the system detects alternative evidence (e.g., *the President*, where *the* is not part of the proper name), it then will block as unsafe the assignment of *The* as a proper name in ambiguous usages of *The President*.

7.2 Frequent-List Lookup Strategy

The frequent-list lookup strategy applies lookup of ambiguously capitalized words in two word lists. The first list contains common words that are frequently found in sentence-starting positions, and the other list contains the most frequent proper names. Both these lists can be compiled completely automatically, as explained in section 5. Thus, if an ambiguous capitalized word is found in the list of frequent sentence-starting common words, it is assigned as a common word, and if it is found in the list of frequent proper names, it is assigned as a proper name. For instance, the word token *The* when used after a period will be recognized as a common word, because *The* is a frequent sentence-starting common word. The Word token *Japan* in a similar context will be recognized as a proper name, because *Japan* is a member of the frequent-proper-name list.

Note, however, that this strategy is applied after the sequence strategy and thus, a word listed in one of the lists will not necessarily be marked according to its list class. The list lookup assignment is applied only to the ambiguously capitalized words that have not been handled by the sequence strategy.

Row B of Table 3 displays the results of the application of the frequent-list lookup strategy to the Brown corpus and the WSJ corpus. The frequent-list lookup strategy produced an error rate of less than 0.5%. A few wrong assignments came from phrases like *Mr. A* and *Mrs. Someone* and words in titles like *I've Got a Dog*, where *A*, *Someone*, and *I* were recognized as common words although they were tagged as proper nouns in the text. The frequent-list lookup strategy is not very effective for proper names, where it covered under 7% of candidates in the Brown corpus and under 3% in the WSJ corpus, but it is extremely effective for common words: it covered over 60% of ambiguous capitalized common words.

7.3 Single-Word Assignment

The sequence strategy is accurate, but it covers only 9–12% of proper names in ambiguous positions. The frequent-list lookup strategy is mostly effective for common words. To boost the coverage on the proper name category, we introduced another DCA strategy. We call this strategy **single-word assignment**, and it can be summarized as follows: if a word in the current document is seen capitalized in an unambiguous position and at the same time it is not used lower-cased anywhere in the document, this word in this particular document is very likely to stand for a proper name even when used capitalized in ambiguous positions. And conversely, if a word in the current document is used only lower-cased (except in ambiguous positions), it is extremely unlikely that this word will act as a proper name in an ambiguous position and thus, such a word can be marked as a common word.

Note that by the time single-word assignment is implemented, the sequence strategy and the frequent-list lookup strategy have been already applied and all high-frequency sentence-initial words have been assigned. This ordering is important, because even if a high-frequency common word is observed in a document only as a proper name (usually as part of a multiword proper name), it is still not safe to mark it as a proper name in ambiguous positions.

Row C of Table 3 displays the results of the application of the single-word assignment strategy to the Brown corpus and the WSJ corpus. The single-word assignment strategy is useful for proper-name identification: although it is not as accurate as the sequence strategy, it still produces a reasonable error rate at the same time boosting the coverage considerably (19–34%). On common words this method is not as effective, with an error rate as high as 6.61% on the WSJ corpus and a coverage below 5%.

The single-word-assignment strategy handles well the so-called unknown-word problem, which arises when domain-specific lexica are missing from a general vocabulary. Since our system is not equipped with a general vocabulary but rather builds a document-specific vocabulary on the fly," important domain-specific words are identified and treated similarly to all other words.

A generally difficult case for the single-word assignment strategy arises when a word is used both as a proper name and as a common word in the same document, especially when one of these usages occurs only in an ambiguous position. For instance, in a document about steel, the only occurrence of *Steel Company* happened to start a sentence. This produced an erroneous assignment of the word *Steel* as a common word. Another example: in a document about *the Acting Judge*, the word *acting* in a sentence *Acting on behalf...* was wrongly classified as a proper name. These difficulties, however, often are compensated for by the sequence strategy, which is applied prior to the single-word assignment strategy and tackles such cases using *n*-grams of words.

7.4 Quotes, Brackets, and "After Abbr." Heuristic

Capitalized words in quotes and brackets do not directly contribute to our primary task of sentence boundary disambiguation, but they still present a case of ambiguity for the task of capitalized-word disambiguation. To tackle them we applied two simple heuristics:

- If a single capitalized word is used in quotes or brackets it is a proper noun (e.g., *John (Cool) Lee*).
- If there is a lowercased word, a number, or a comma that is followed by an opening bracket and then by a capitalized word, this capitalized word is a proper noun (e.g., *... happened (Moscow News reported yesterday) but...*).

These heuristics are reasonably accurate: they achieved under 2% error rate on our two test corpora, but they covered only about 6–7% of proper names.

When we studied the distribution of capitalized words after capitalized abbreviations, we uncovered an interesting empirical fact. A capitalized word that follows a capitalized abbreviation is almost certainly a proper name unless it is listed in the list of frequent sentence-starting common words (i.e., it is not *The*, *However*, etc.). The error rate of this heuristic is about 0.8% and, not surprisingly, in 99.5% of cases the abbreviation and the following proper name belonged to the same sentence. Naturally, the coverage of this "after abbr." heuristic depends on the proportion of capitalized abbreviations in the text. In our two corpora this heuristic disambiguated about 20% of ambiguous capitalized proper names.

7.5 Tagging Proper Names: The Overall Performance

In general, the cascading application of the above-described strategies achieved an error rate of about 1%, but it left unclassified about 9% of ambiguous capitalized words in the Brown corpus and 15% of such words in the WSJ corpus. Row D of Table 3 displays the results of the application of the cascading application of the capitalized-word disambiguation strategies to the Brown corpus and the WSJ corpus.

For the proper-name category, the most productive strategy was single-word assignment, followed by the "after abbr." strategy, and then the sequence strategy. For common words, the most productive was the frequent-list lookup strategy, followed by the sequence strategy.

Since our system left unassigned 10–15% of ambiguous capitalized words, we have to decide what to do with them. To keep our system simple and domain independent, we opted for the lexical-lookup strategy that we evaluated in Section 3. This strategy, of course, is not very accurate, but it is applied only to the unassigned words. Row E of Table 3 displays the results of applying the lexical-lookup strategy after the DCA methods. We see that the error rate went up in comparison to the DCA-only method by more than three times (2.9% on the Brown corpus and 4.9% on the WSJ corpus), but no unassigned ambiguous capitalized words are left in the text.

8. Putting It All Together: Assigning Sentence Boundaries

After abbreviations have been identified and capitalized words have been classified into proper names and common words, the system can carry out the assignments of sentence boundaries using the SBD rule set described in Section 3 and listed in Appendix A. This rule set makes use of the observation that if we have at our disposal unambiguous (but not necessarily correct) information as to whether a particular word that precedes a period is an abbreviation and whether the word that follows this period is a proper name, then in mixed-case texts we can easily assign a period (and other potential sentence termination punctuation) as a sentence break or not.

The only ambiguous outcome is generated by the configuration in which an abbreviation is followed by a proper name. We decided to handle this case by applying a crude and simple strategy of always resolving it as "not sentence boundary." On one hand, this makes our method simple and robust, but on the other hand, it imposes some penalty on its performance.

Row A of Table 4 summarizes the upper bound for our SBD approach: when we have entirely correct information on the abbreviations and proper names, as explained in Section 3.1. There the erroneous assignments come only from the crude treatment of abbreviations that are followed by proper names.

Table 4
Error rates measured on the SBD, capitalized-word disambiguation, and abbreviation identification tasks achieved by different methods described in this article.

Method	Brown Corpus			WSJ Corpus		
	SBD	Capitalized words	Abbreviations	SBD	Capitalized words	Abbreviations
A Upper bound	0.01%	0.0%	0.0%	0.13%	0.0%	0.0%
B Lower bound	2.00%	7.40%	10.8%	4.10%	15.0%	9.6%
C Best quoted	0.20%	3.15%	—	0.50%	4.72%	—
D **DCA**	**0.28%**	**2.83%**	**0.8%**	**0.45%**	**4.88%**	**1.2%**
E DCA (no abbreviations lexicon)	0.65%	2.89%	8.9%	1.41%	4.92%	6.6%
F POS tagger	0.25%	3.15%	1.2%	0.39%	4.72%	2.1%
G POS tagger + DCA	0.20%	1.87%	0.8%	0.31%	3.22%	1.2%

Row B of Table 4 summarizes the lower-bound results. The lower bound for our approach was estimated by applying the lexical-lookup strategy for capitalized-word disambiguation together with the abbreviation-guessing heuristics to feed the SBD rule set, as described in Section 3.2. Here we see a significant impact of the infelicities in the disambiguation of capitalized words and abbreviations on the performance of the SBD rule set.

Row C of Table 4 summarizes the highest results known to us (for all three tasks) produced by automatic systems on the Brown corpus and the WSJ corpus. State-of-the-art machine learning and rule-based SBD systems achieve an error rate of 0.8–1.5% measured on the Brown corpus and the WSJ corpus. The best performance on the WSJ corpus was achieved by a combination of the SATZ system (Palmer and Hearst 1997) with the Alembic system (Aberdeen et al. 1995): a 0.5% error rate. The best performance on the Brown corpus, a 0.2% error rate, was reported by Riley (1989), who trained a decision tree classifier on a 25-million-word corpus. In the disambiguation of capitalized words, the most widespread method is POS tagging, which achieves about a 3% error rate on the Brown corpus and a 5% error rate on the WSJ corpus, as reported in Mikheev (2000). We are not aware of any studies devoted to the identification of abbreviations with comprehensive evaluation on either the Brown corpus or the WSJ corpus.

In row D of Table 4, we summarized our main results: the results obtained by the application of our SBD rule set, which uses the information provided by the DCA to capitalized word disambiguation applied together with lexical lookup (as described in Section 7.5), and the abbreviation-handling strategy, which included the guessing heuristics, the DCA, and the list of 270 abbreviations (as described in Section 6). As can be seen in the table, the performance of this system is almost indistinguishable from the best previously quoted results. On proper-name disambiguation, it achieved a 2.83% error rate on the Brown corpus and a 4.88% error rate on the WSJ corpus. On the SBD task, it achieved a 0.28% error rate on the Brown corpus and a 0.45% error rate on the WSJ corpus. If we compare these results with the upper bound for our SBD approach, we can see that the infelicities in proper-name and abbreviation identification introduced an increase of about 0.3% in the error rate on the SBD task.

To test the adaptability of our approach to a completely new domain, we applied our system in a configuration in which it was *not* equipped with the list of 270 abbre-

viations, since this list is the only domain-sensitive resource in our system. The results for this configuration are summarized in row E of Table 4. The error rate increase of 5–7% on the abbreviation handling introduced about a twofold increase in the SBD error rate on the Brown corpus (a 0.65% error rate) and about a threefold increase on the WSJ corpus (1.41%). But these results are still comparable to those of the majority of currently used sentence splitters.

9. Detecting Limits for the DCA

Since our DCA method relies on the assumption that the words it tries to disambiguate occur multiple times in a document, its performance clearly should depend on the length of the document: very short documents possibly do not provide enough disambiguation clues, whereas very long documents possibly contain too many clues that cancel each other.

As noted in Section 2.1, the average length of the documents in the Brown corpus is about 2,300 words. Also, the documents in that corpus are distributed very densely around their mean. Thus not much can be inferred about the dependency of the performance of the method on document length apart from the observation that documents 2,000–3,000 words long are handled well by our approach. In the WSJ corpus, the average length of the document is about 500 words, and therefore we could investigate the effect of short documents on the performance. We divided documents into six groups according to their length and plotted the error rate for the SBD and capitalized-word disambiguation tasks as well as the number of documents in a group, as shown in Figure 2. As can be seen in the figure, short documents (50 words or less) have the highest average error rate both for the SBD task (1.63) and for the capitalized-word disambiguation task (5.25). For documents 50 to 100 words long, the error rate is still a bit higher than normal, and for longer documents the error rate stabilizes around 1.5 for the capitalized-word disambiguation task and 0.3 for the SBD task. The error rate on documents 2,000 words long and higher is almost identical to that registered on the Brown corpus on documents of the same length.

Thus here we can conclude that the proposed approach tends not to be very effective for documents shorter than 50 words (one to three sentences), but it handles well documents up to 4,000 words long. Since our corpora did not contain documents significantly longer than that, we could not estimate whether or when the performance of our method significantly deteriorates on longer documents. We also evaluated the performance of the method on different subcorpora of the Brown corpus: the most difficult subdomains proved to be scientific texts, spoken-language transcripts, and journalistic texts, whereas fiction was the easiest genre for the system.

10. Incorporating DCA into a POS Tagger

To test our hypothesis that DCA can be used as a complement to a local-context approach, we combined our main configuration (evaluated in row D of Table 4) with a POS tagger. Unlike other POS taggers, this POS tagger (Mikheev 2000) was also trained to disambiguate sentence boundaries.

10.1 Training a POS Tagger

In our markup convention (Section 2), periods are tokenized as separate tokens regardless of whether they stand for fullstops or belong to abbreviations. Consequently a POS tagger can naturally treat them similarly to any other ambiguous words. There is, however, one difference in the implementation of such a tagger. Normally, a POS

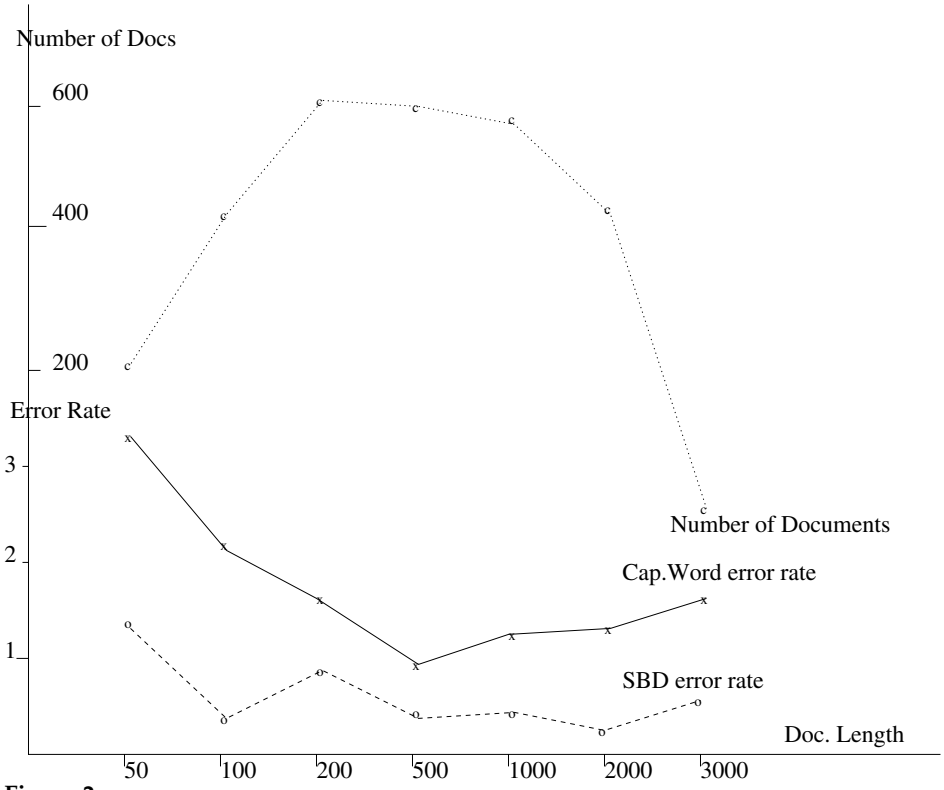

Figure 2
Distribution of the error rate and the number of documents across the document length (measured in word tokens) in the WSJ corpus.

tagger operates on text spans that form a sentence. This requires resolving sentence boundaries before tagging. We see no good reason, however, why such text spans should necessarily be sentences, since the majority of tagging paradigms (e.g., Hidden Markov Model [HMM] [Kupiec 1992], Brill's [Brill 1995a], and MaxEnt [Ratnaparkhi 1996]) do not attempt to parse an entire sentence and operate only in the local window of two to three tokens. The only reason why taggers traditionally operate on the sentence level is that a sentence naturally represents a text span in which POS information does not depend on the previous and following history.

This issue can be also addressed by breaking the text into short text spans at positions where the previous tagging history does not affect current decisions. For instance, a bigram tagger operates within a window of two tokens, and thus a sequence of word tokens can be terminated at an unambiguous word token, since this unambiguous word token will be the only history used in tagging of the next token. At the same time since this token is unambiguous, it is not affected by the history. A trigram tagger operates within a window of three tokens, and thus a sequence of word tokens can be terminated when two unambiguous words follow each other.

Using Penn Treebank with our tokenization convention (Section 2), we trained a trigram HMM POS tagger. Words were clustered into ambiguity classes (Kupiec 1992) according to the sets of POS tags they can take on. The tagger predictions were based on the ambiguity class of the current word, abbreviation/capitalization information,

and trigrams of POS tags:

$$P(t_1 \dots t_n O_1 \dots O_n) = \text{argmax} \prod_{i=1}^{i=n} P(O_i \mid t_i) * P(t_i \mid t_{i-1} t_{i-2} a_{i-1})$$

where t_i is a disambiguated POS tag of the ith word, a_i is the abbreviation flag of the ith word, and O_i is the observation at the ith position, which in our case is the ambiguity class the word belongs to, its capitalization, and its abbreviation flag ($AmbClass_i, a_i, Cap_i$). Since the abbreviation flag of the previous word strongly influences period disambiguation, it was included in the standard trigram model.

We decided to train the tagger with the minimum of preannotated resources. First, we used 20,000 tagged words to "bootstrap" the training process, because purely unsupervised techniques, at least for the HMM class of taggers, yield lower precision. We also used our DCA system to assign capitalized words, abbreviations, and sentence breaks, retaining only cases assigned by the strategies with an accuracy not less than 99.8%. This was done because purely unsupervised techniques (e.g., Baum-Welch [Baum and Petrie 1966] or Brill's [Brill 1995b]) enable regularities to be induced for word classes which contain many entries, exploiting the fact that individual words that belong to a POS class occur in different ambiguity patterns. Counting all possible POS combinations in these ambiguity patterns over multiple patterns usually produces the right combinations as the most frequent. Periods as many other closed-class words cannot be successfully covered by such technique.

After bootstrapping we applied the forward-backward (Baum-Welch) algorithm (Baum and Petrie 1966) and trained our tagger in the *unsupervised* mode, that is, without using the annotation available in the Brown corpus and the WSJ corpus. For evaluation purposes we trained (and bootstrapped) our tagger on the Brown corpus and applied it to the WSJ corpus and vice versa. We preferred this method to tenfold cross-validation because it allowed us to produce only two tagging models instead of twenty and also enabled us to test the tagger in harsher conditions, that is, when it is applied to texts that are very distant from the ones on which it was trained.

The overall performance of the tagger was close to 96%, which is a bit lower than the best quoted results. This can be accounted for by the fact that training and evaluation were performed on two very different text corpora, as explained above. The performance of the tagger on our target categories (periods and proper names) was very close to that of the DCA method, as can be seen in row F of Table 4.

10.2 POS Tagger and the DCA

We felt that the DCA method could be used as a complement to the POS tagger, since these techniques employ different types of information: in-document distribution and local context. Thus, a hybrid system can deliver at least two advantages. First, 10–15% of the ambiguous capitalized words unassigned by the DCA can be assigned using a standard POS-tagging method based on the local syntactic context rather than the inaccurate lexical-lookup approach. Second, the local context can correct some of the errors made by the DCA.

To implement this hybrid approach we incorporated the DCA system into the POS tagger. We modified the tagger model by incorporating the DCA predictions using linear interpolation:

$$P(combined) = \lambda * P(tagger) + (1 - \lambda) * P(DCA\_Strategy)$$

where $P(DCA\_Strategy)$ is the accuracy of a specific DCA strategy and $P(tagger)$ is the probability assigned by the tagger's model. Although it was possible to estimate an

optimal value for λ from the tagged corpus, we decided simply to set it to be 0.5 (i.e., giving similar weight to both sources of information). Instead of using the SBD rule set described in Section 3, in this configuration, period assignments were handled by the tagger's model.

Row G of Table 4 displays the results of the application of the hybrid system. We see an improvement on proper-name recognition in comparison to the DCA or POS-tagging approaches (rows D and F) by about a 30–40% cut in the error rate: an overall error rate of 1.87% on the Brown corpus and of 3.22% on the WSJ corpus. In turn this enabled better tagging of sentence boundaries: a 0.20% error rate on the Brown corpus and a 0.31% error rate on the WSJ corpus, which corresponds to about a 20% cut in the error rate in comparison to the DCA or the POS-tagging approaches alone.

Thus, although for applications that rely on POS tagging it probably makes more sense to have a single system that assigns both POS tags and sentence boundaries, there is still a clear benefit in using the DCA method because

- the DCA method incorporated into the POS tagger significantly reduced the error rate on the target categories (periods and proper names).
- the DCA method is domain independent, whereas taggers usually need to be trained for each specific domain to obtain best results.
- the DCA system was used in resource preparation for training the tagger.
- the DCA system is significantly faster than the tagger, does not require resource development, and for tasks that do not require full POS information, it is a preferable solution.

So in general, the DCA method can be seen as an enhancer for a POS tagger and also as a lightweight alternative to such a tagger when full POS information is not required.

11. Further Experiments

11.1 The Cache Extension

One of the features of the method advocated in this article is that the system collects suggestive instances of usage for target words from each document, then applies this information during the second pass through the document (actual processing), and then "forgets" what it has learned before handling another document. The main reason for not carrying over the information that has been inferred from one document to process another document is that in general we do not know whether this new document comes from the same corpus as the first document, and thus the regularities that have been identified in the first document might not be useful, but rather harmful, when applied to that new document. When we are dealing with documents of reasonable length, this "forgetful" behavior does not matter much, because such documents usually contain enough disambiguation clues. As we showed in Section 8, however, when short documents of one to three sentences are being processed, quite often there are not enough disambiguation clues within the document itself, which leads to inferior performance.

To improve the performance on short documents, we introduced a special caching module that propagates some information identified in previously processed documents to the processing of a new one. To propagate features of individual words from one document to processing another one is a risky strategy, since words are very

ambiguous. Word sequences, however, are much more stable and can be propagated across documents. We decided to accumulate in our cache all multiword proper names and lower-cased word bigrams induced by the sequence strategy (Section 7.1). These word sequences are used by the sequence strategy exactly as are word sequences induced on the fly, and then the induced on-the-fly sequences are added to the cache. We also add to the cache the bigrams of abbreviations and regular words induced by the abbreviation-handling module, as explained in Section 6. These bigrams are used together with the bigrams induced on the fly. This strategy proved to be quite useful: it covered another 2% of unresolved cases (before applying the lexical lookup), with an error rate of less than 1%.

11.2 Handling Russian News

To test how easy it is to apply the DCA to a new language, we tested it on a corpus of British Broadcasting Corporation (BBC) news in Russian. We collected this corpus from the Internet ⟨*http://news.bbc.co.uk/hi/russian/world/default.htm*⟩ over a period of 30 days. This gave us a corpus of 300 short documents (one or two paragraphs each). We automatically created the supporting resources from 364,000 documents from the Russian corpus of the European Corpus Initiative, using the method described in section 5.

Since, unlike English, Russian is a highly inflected language, we had to deal with the case normalization issue. Before using the DCA method, we applied a Russian morphological processor (Mikheev and Liubushkina 1995) to convert each word in the text to its main form: nominative case singular for nouns and adjectives, infinitive for verbs, etc. For words that could be normalized to several main forms (polysemy), when secondary forms of different words coincided, we retained all the main forms. Since the documents in the BBC news corpus were rather short, we applied the cache module, as described in Section 11.1. This allowed us to reuse information across the documents.

Russian proved to be a simpler case than English for our tasks. First, on average, Russian words are longer than English words: thus the identification of abbreviations is simpler. Second, proper names in Russian coincide less frequently with common words; this makes the disambiguation of capitalized words in ambiguous positions easier. The overall performance reached a 0.1% error rate on sentence boundaries and a 1.8% error rate on ambiguous capitalized words, with the coverage on both tasks at 100%.

12. Related Research

12.1 Research in Nonlocal Context

The use of nonlocal context and dynamic adaptation have been studied in language modeling for speech recognition. Kuhn and de Mori (1998) proposed a cache model that works as a kind of short-term memory by which the probability of the most recent n words is increased over the probability of a general-purpose bigram or trigram model. Within certain limits, such a model can adapt itself to changes in word frequencies, depending on the topic of the text passage. The DCA system is similar in spirit to such dynamic adaptation: it applies word n-grams collected on the fly from the document under processing and favors them more highly than the default assignment based on prebuilt lists. But unlike the cache model, it uses a multipass strategy.

Clarkson and Robinson (1997) developed a way of incorporating standard n-grams into the cache model, using mixtures of language models and also exponentially decaying the weight for the cache prediction depending on the recency of the word's last

occurrence. In our experiments we applied simple linear interpolation to incorporate the DCA system into a POS tagger. Instead of decaying nonlocal information, we opted for not propagating it from one document for processing of another. For handling very long documents with our method, however, the information decay strategy seems to be the right way to proceed.

Mani and MacMillan (1995) pointed out that little attention had been paid in the named-entity recognition field to the discourse properties of proper names. They proposed that proper names be viewed as linguistic expressions whose interpretation often depends on the discourse context, advocating text-driven processing rather than reliance on pre-existing lists. The DCA outlined in this article also uses nonlocal discourse context and does not heavily rely on pre-existing word lists. It has been applied not only to the identification of proper names, as described in this article, but also to their classification (Mikheev, Grover, and Moens 1998).

Gale, Church, and Yarowsky (1992) showed that words strongly tend to exhibit only one sense in a document or discourse ("one sense per discourse"). Since then this idea has been applied to several tasks, including word sense disambiguation (Yarowsky 1995) and named-entity recognition (Cucerzan and Yarowsky 1999). Gale, Church, and Yarowsky's observation is also used in our DCA, especially for the identification of abbreviations. In capitalized-word disambiguation, however, we use this assumption with caution and first apply strategies that rely not just on single words but on words together with their local contexts (n-grams). This is similar to "one sense per collocation" idea of Yarowsky (1993).

The description of the EAGLE workbench for linguistic engineering (Baldwin et al. 1997) mentions a case normalization module that uses a heuristic in which a capitalized word in an ambiguous position should be rewritten without capitalization if it is found lower-cased in the same document. This heuristic also employs a database of bigrams and unigrams of lower-cased and capitalized words found in unambiguous positions. It is quite similar to our method for capitalized-word disambiguation. The description of the EAGLE case normalization module provided by Baldwin et al. is, however, very brief and provides no performance evaluation or other details.

12.2 Research in Text Preprocessing

12.2.1 Sentence Boundary Disambiguation. There exist two large classes of SBD systems: rule based and machine learning. The rule-based systems use manually built rules that are usually encoded in terms of regular-expression grammars supplemented with lists of abbreviations, common words, proper names, etc. To put together a few rules is fast and easy, but to develop a rule-based system with good performance is quite a labor-consuming enterprise. For instance, the Alembic workbench (Aberdeen et al. 1995) contains a sentence-splitting module that employs over 100 regular-expression rules written in Flex. Another well-acknowledged shortcoming of rule-based systems is that such systems are usually closely tailored to a particular corpus or sublanguage and are not easily portable across domains.

Automatically trainable software is generally seen as a way of producing systems that are quickly retrainable for a new corpus, for a new domain, or even for another language. Thus, the second class of SBD systems employs machine learning techniques such as decision tree classifiers (Riley 1989), neural networks (Palmer and Hearst 1994), and maximum-entropy modeling (Reynar and Ratnaparkhi 1997). Machine learning systems treat the SBD task as a classification problem, using features such as word spelling, capitalization, suffix, and word class found in the local context of a potential sentence-terminating punctuation sign. Although training of such

systems is completely automatic, the majority of machine learning approaches to the SBD task require labeled examples for training. This implies an investment in the data annotation phase.

The main difference between the existing machine learning and rule-based methods for the SBD task and our approach is that we decomposed the SBD task into several subtasks. We decided to tackle the SBD task through the disambiguation of the period preceding and following words and then feed this information into a very simple SBD rule set. In contrast, the standard practice in building SBD software is to disambiguate configurations of a period with its ambiguous local context in a single step, either by encoding disambiguation clues into the rules or inferring a classifier that accounts for the ambiguity of the words on the left and on the right of the period.

Our approach to SBD is closer in spirit to machine learning methods because its retargeting does not require rule reengineering and can be done completely automatically. Unlike traditional machine learning SBD approaches, however, our approach does not require annotated data for training.

12.2.2 Disambiguation of Capitalized Words. Disambiguation of capitalized words is usually handled by POS taggers, which treat capitalized words in the same way as other categories, that is, by accounting for the immediate syntactic context and using estimates collected from a training corpus. As Church (1988) rightly pointed out, however, "Proper nouns and capitalized words are particularly problematic: some capitalized words are proper nouns and some are not. Estimates from the Brown Corpus can be misleading. For example, the capitalized word 'Acts' is found twice in the Brown Corpus, both times as a proper noun (in a title). It would be misleading to infer from this evidence that the word 'Acts' is always a proper noun."

In the information extraction field, the disambiguation of ambiguous capitalized words has always been tightly linked to the classification of proper names into semantic classes such as person name, location, and company name. Named-entity recognition systems usually use sets of complex hand-crafted rules that employ a gazetteer and a local context (Krupa and Hausman 1998). In some systems such dependencies are learned from labeled examples (Bikel et al. 1997). The advantage of the named-entity approach is that the system not only identifies proper names but also determines their semantic class. The disadvantage is in the cost of building a wide-coverage set of contextual clues manually or producing annotated training data. Also, the contextual clues are usually highly specific to the domain and text genre, making such systems very difficult to port.

Both POS taggers and named-entity recognizers are normally built using the local-context paradigm. In contrast, we opted for a method that relies on the entire distribution of a word in a document. Although it is possible to train some classes of POS taggers without supervision, this usually leads to suboptimal performance. Thus the majority of taggers are trained using at least some labeled data. Named-entity recognition systems are usually hand-crafted or trained from labeled data. As was shown above, our method does not require supervised training.

12.2.3 Disambiguation of Abbreviations. Not much information has been published on abbreviation identification. One of the better-known approaches is described in Grefenstette and Tapanainen (1994), which suggested that abbreviations first be extracted from a corpus using abbreviation-guessing heuristics akin to those described in Section 6 and then reused in further processing. This is similar to our treatment of abbreviation handling, but our strategy is applied on the document rather than corpus level. The main reason for restricting abbreviation discovery to a single document is

that this does not presuppose the existence of a corpus in which the current document is similar to other documents.

Park and Byrd (2001) recently described a hybrid method for finding abbreviations and their definitions. This method first applies an "abbreviation recognizer" that generates a set of "candidate abbreviations" for a document. Then for this set of candidates the system tries to find in the text their definitions (e.g., *United Kingdom* for *UK*). The abbreviation recognizer for these purposes is allowed to overgenerate significantly. There is no harm (apart from the performance issues) in proposing too many candidate abbreviations, because only those that can be linked to their definitions will be retained. Therefore the abbreviation recognizer treats as a candidate any token of two to ten characters that contains at least one capital letter. Candidates then are filtered through a set of known common words and proper names. At the same time many good abbreviations and acronyms are filtered out because not for all of them will definitions exist in the current document.

In our task we are interested in finding all and only abbreviations that end with a period (proper abbreviations rather than acronyms), regardless of whether they can be linked to their definitions in the current document or not. Therefore, in our method we cannot tolerate candidate overgeneration or excessive filtering and had to develop more selective methods for finding abbreviations in text.

13. Discussion

In this article we presented an approach that tackles three important aspects of text normalization: sentence boundary disambiguation, disambiguation of capitalized words when they are used in positions where capitalization is expected, and identification of abbreviations. The major distinctive features of our approach can be summarized as follows:

- We tackle the sentence boundary task only after we have fully disambiguated the word on the left and the word on the right of a potential sentence boundary punctuation sign.
- To disambiguate capitalized words and abbreviations, we use information distributed across the entire document rather than their immediate local context.
- Our approach does not require manual rule construction or data annotation for training. Instead, it relies on four word lists that can be generated completely automatically from a raw (unlabeled) corpus.

In this approach we do not try to resolve each ambiguous word occurrence individually. Instead, the system scans the entire document for the contexts in which the words in question are used unambiguously, and this gives it grounds, acting by analogy, for resolving ambiguous contexts.

We deliberately shaped our approach so that it largely does not rely on precompiled statistics, because the most interesting events are inherently infrequent and hence are difficult to collect reliable statistics for. At the same time precompiled statistics would be smoothed across multiple documents rather than targeted to a specific document. By collecting suggestive instances of usage for target words from each particular document on the fly, rather than relying on preacquired resources smoothed across the entire document collection, our approach is robust to domain shifts and new lexica and closely targeted to each document.

A significant advantage of this approach is that it can be targeted to new domains completely automatically, without human intervention. The four word lists that our system uses for its operation can be generated automatically from a raw corpus and require no human annotation. Although some SBD systems can be trained on relatively small sets of labeled examples, their performance in such cases is somewhat lower than their optimal performance. For instance, Palmer and Hearst (1997) report that the SATZ system (decision tree variant) was trained on a set of about 800 labeled periods, which corresponds to a corpus of about 16,000 words. This is a relatively small training set that can be manually marked in a few hours' time. But the error rate (1.5%) of the decision tree classifier trained on this small sample was about 50% higher than that when trained on 6,000 labeled examples (1.0%).

The performance of our system does not depend on the availability of labeled training examples. For its "training," it uses a raw (unannotated in any way) corpus of texts. Although it needs such a corpus to be relatively large (a few hundred thousand words), this is normally not a problem, since when the system is targeted to a new domain, such a corpus is usually already available at no extra cost. Therefore there is no trade-off between the amount of human labor and the performance of the system. This not only makes retargeting of such system easier but also enables it to be operational in a completely autonomous way: it needs only to be pointed to texts from a new domain, and then it can retarget itself automatically.

Although the DCA requires two passes through a document, the simplicity of the underlying algorithms makes it reasonably fast. It processes about 3,000 words per second using a Pentium II 400 MHz processor. This includes identification of abbreviations, disambiguation of capitalized words, and then disambiguation of sentence boundaries. This is comparable to the speed of other preprocessing systems.[3] The operational speed is about 10% higher than the training speed because, apart from applying the system to the training corpus, training also involves collecting, thresholding, and sorting of the word lists—all done automatically but at extra time cost. Training on the 300,000-word NYT text collection took about two minutes.

Despite its simplicity, the performance of our approach was on the level with the previously highest reported results on the same test collections. The error rate on sentence boundaries in the Brown corpus was not significantly worse than the lowest quoted before (Riley 1989: 0.28% vs. 0.20% error rate). On the WSJ corpus our system performed slightly better than the combination of the Alembic and SATZ systems described in Palmer and Hearst (1997) (0.44% vs. 0.5% error rate). Although these error rates seem to be very small, they are quite significant. Unlike general POS tagging, in which it is unfair to expect an error rate of less than 2% because even human annotators have a disagreement rate of about 3%, sentence boundaries are much less ambiguous (with a disagreement of about 1 in 5,000). This shows that an error rate of 1 in 200 (0.5%) is still far from reaching the disagreement level. On the other hand, one error in 200 periods means that there is one error in every two documents in the Brown corpus and one error in every four documents in the WSJ corpus.

With all its strong points, there are a number of restrictions to the proposed approach. First, in its present form it is suitable only for processing of reasonably "well-behaved" texts that consistently use capitalization (mixed case) and do not contain much noisy data. Thus, for instance, we do not expect our system to perform well on single-cased texts (e.g., texts written in all capital or all lower-cased letters) or on

3 Palmer and Hearst (1997) report a speed of over 10,000 sentences a minute, which with their average sentence length of 20 words equals over 3,000 words per second, but on a slower machine (DEC Alpha 3000).

optical character reader–generated texts. We noted in Section 8 that very short documents of one to three sentences also present a difficulty for our approach. This is where robust syntactic systems like SATZ (Palmer and Hearst 1997) or the POS tagger reported in Mikheev (2000), which do not heavily rely on word capitalization and are not sensitive to document length, have an advantage.

Our DCA uses information derived from the entire document and thus can be used as a complement to approaches based on the local context. When we incorporated the DCA system into a POS tagger (Section 8), we measured a 30–35% cut in the error rate on proper-name identification in comparison to DCA or the POS-tagging approaches alone. This in turn enabled better tagging of sentence boundaries: a 0.20% error rate on the Brown corpus and a 0.31% error rate on the WSJ corpus, which corresponds to about a 20% cut in the error rate in comparison to DCA or the POS-tagging approaches alone.

We also investigated the portability of our approach to other languages and obtained encouraging results on a corpus of news in Russian. This strongly suggests that the DCA method can be applied to the majority of European languages, since they share the same principles of capitalization and word abbreviation. Obvious exceptions, though, are German and some Scandinavian languages in which capitalization is used for things other than proper-name and sentence start signaling. This does not mean, however, that the DCA in general is not suitable for preprocessing of German texts—it just needs to be applied with different disambiguation clues.

Initially the system described in this article was developed as a text normalization module for a named-entity recognition system (Mikheev, Grover, and Moens 1998) that participated in MUC-7. There the ability to identify proper names with high accuracy proved to be instrumental in enabling the entire system to achieve a very high level of performance. Since then this text normalization module has been used in several other systems, and its ability to be adapted easily to new domains enabled rapid development of text analysis capabilities in medical, legal, and law enforcement domains.

Appendix A: SBD Rule Set

In this section we present the rule set used by our system to assign potential sentence boundary punctuation as

FS	Punctuation that signals end of sentence
AP	Period that is part of abbreviation
AFS	Period that is part of abbreviation and signals end of sentence

This rule set operates over tokens that are disambiguated as to whether or not they are abbreviations and whether or not they are proper names. Tokens are categorized into overlapping sets as follows:

NONE	No token (end of input)
ANY	Any token
ANY-OR-NONE	Any token or no token at all
ABBR	Token that was disambiguated as “abbreviation” (Note: . . . Ellipsis is treated as an abbreviation too)
Not_ABBR	Nonpunctuation token that was disambiguated as “not abbreviation”
CLOSE_PUNCT	Closing quotes, closing brackets
OPEN_PUNCT	Opening quotes, opening brackets

PUNCT	Punctuation token not CLOSE_PUNCT or OPEN_PUNCT or [.!?;]
NUM	Number
LOW_COMMON	Lower-cased common word
CAP_COMMON	Capitalized word that was disambiguated as a common word
CAP_PROP	Capitalized word that was disambiguated as a proper name
PROPER_NAME	Proper name

Rule Set

word-2	word-1	FOCAL	word+1	word+2	Assign	Example
ANY	Not_ABBR	[.?!]	ANY-OR-NONE	ANY-OR-NONE	FS	book.
ANY	CLOSE_PUNCT	[.?!]	ANY-OR-NONE	ANY-OR-NONE	FS	).
ABBR	.	[.?!]	ANY-OR-NONE	ANY-OR-NONE	FS	Tex.!
ANY	ANY	;	CAP_COMMON	ANY-OR-NONE	FS	; The
	ABBR	.	NONE	NONE	AFS	Tex.EOF
	ABBR	.	CAP_COMMON	ANY-OR-NONE	AFS	Tex. The
	ABBR	.	CLOSE_PUNCT	CAP_COMMON	AFS	kg.) This
	ABBR	.	OPEN_PUNCT	CAP_COMMON	AFS	kg. (This
	ABBR	.	CLOSE_PUNCT OPEN_PUNCT	CAP_COMMON	AFS	kg.) (This
	ABBR	.	PUNCT	ANY-OR-NONE	AP	kg.,
	ABBR	.	[.?!]	ANY-OR-NONE	AP	Tex.!
	ABBR	.	LOW_COMMON	ANY-OR-NONE	AP	kg. this
	ABBR	.	CLOSE_PUNCT	LOW_COMMON	AP	kg.) this
	ABBR	.	OPEN_PUNCT	LOW_COMMON	AP	kg. (this
	ABBR	.	CLOSE_PUNCT OPEN_PUNCT	LOW_COMMON	AP	kg.) (this
	ABBR	.	ABBR	.	AP	Sen. Gen.
	ABBR	.	NUM	ANY-OR-NONE	AP	kg. 5
	ABBR	.	PROPER_NAME	ANY-OR-NONE	AP	Dr. Smith

Acknowledgments

The work reported in this article was supported in part by grant GR/L21952 (Text Tokenization Tool) from the Engineering and Physical Sciences Research Council, U.K., and also it benefited from the ongoing efforts in building domain-independent text-processing software at Infogistics Ltd. I am also grateful to one anonymous reviewer who put a lot of effort into making this article as it is now.

References

Aberdeen, John S., John D. Burger, David S. Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain. 1995. "Mitre: Description of the alembic system used for MUC-6." In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland, November. Morgan Kaufmann.

Baldwin, Breck, Christine Doran, Jeffrey Reynar, Michael Niv, Bangalore Srinivas, and Mark Wasson. 1997. "EAGLE: An extensible architecture for general linguistic engineering." In *Proceedings of Computer-Assisted Information Searching on Internet (RIAO '97)*, Montreal, June.

Baum, Leonard E. and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite Markov chains. *Annals of Mathematical Statistics* 37:1559–1563.

Bikel, Daniel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. "Nymble: A high performance learning name-finder." In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP'97)*, pages 194–200. Washington, D.C., Morgan Kaufmann.

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

Brill, Eric. 1995b. "Unsupervised learning of disambiguation rules for part of speech tagging." In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 1–13, Somerset, New Jersey. Association for Computational Linguistics.

Burnage, Gavin. 1990. *CELEX: A Guide for Users*. Centre for Lexical Information, Nijmegen, Netherlands.

Chinchor, Nancy. 1998. "Overview of MUC-7." In *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax*, April. Morgan Kaufmann.

Church, Kenneth. 1988. "A stochastic parts program and noun-phrase parser for unrestricted text." In *Proceedings of the Second ACL Conference on Applied Natural Language Processing (ANLP'88)*, pages 136–143, Austin, Texas.

Church, Kenneth. 1995. "One term or two?" In *SIGIR'95, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 310–318, Seattle, Washington, July. ACM Press.

Clarkson, Philip and Anthony J. Robinson. 1997. "Language model adaptation using mixtures and an exponentially decaying cache." In *Proceedings IEEE International Conference on Speech and Signal Processing*, Munich, Germany.

Cucerzan, Silviu and David Yarowsky. 1999. "Language independent named entity recognition combining morphological and contextual evidence." In *Proceedings of Joint SIGDAT Conference on EMNLP and VLC*.

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

Gale, William, Kenneth Church, and David Yarowsky. 1992. "One sense per discourse." In *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*, pages 233–237.

Grefenstette, Gregory and Pasi Tapanainen. 1994. "What is a word, what is a sentence? Problems of tokenization." In *The Proceedings of Third Conference on Computational Lexicography and Text Research (COMPLEX'94)*, Budapest, Hungary.

Krupka, George R. and Kevin Hausman. 1998. Isoquest Inc.: Description of the netowl extractor system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA. Morgan Kaufmann.

Kuhn, Roland and Renato de Mori. 1998. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12:570–583.

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

Mani, Inderjeet and T. Richard MacMillan. 1995. "Identifying unknown proper names in newswire text." In B. Boguraev and J. Pustejovsky, editors, *Corpus Processing for Lexical Acquisition*. MIT Press, Cambridge, Massachusetts, pages 41–59.

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

Mikheev, Andrei. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics* 23(3):405–423.

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

Mikheev, Andrei. 2000. "Tagging sentence boundaries." In *Proceedings of the First Meeting of the North American Chapter of the Computational Linguistics (NAACL'2000)*, pages 264–271, Seattle, Washington. Morgan Kaufmann.

Mikheev, Andrei, Clair Grover, and Colin Matheson. 1998. *TTT: Text Tokenisation Tool*. Language Technology Group, University of Edinburgh. Available at http://www.ltg.ed.ac.uk/software/ttt/index.html.

Mikheev, Andrei, Clair Grover, and Marc Moens. 1998. Description of the ltg system used for MUC-7. In *Seventh Message Understanding Conference (MUC–7): Proceedings of a Conference Held in Fairfax*, Virginia. Morgan Kaufmann.

Mikheev, Andrei and Liubov Liubushkina. 1995. Russian morphology: An engineering approach. *Natural Language Engineering* 1(3):235–260.

Palmer, David D. and Marti A. Hearst. 1994. "Adaptive sentence boundary disambiguation." In *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing (ANLP'94)*, pages 78–83, Stuttgart, Germany, October. Morgan Kaufmann.

Palmer, David D. and Marti A. Hearst. 1997. Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics* 23(2):241–269.

Park, Youngja and Roy J. Byrd. 2001. "Hybrid text mining for finding abbreviations and their definitions." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMLP'01)*, pages 16–19, Washington, D.C. Morgan Kaufmann.

Ratnaparkhi, Adwait. 1996. "A maximum entropy model for part-of-speech

tagging." In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania, Philadelphia.

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 (ANLP'97)*, pages 16–19. Morgan Kaufmann.

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. Morgan Kaufmann.

Yarowsky, David. 1993. "One sense per collocation." In *Proceedings of ARPA Human Language Technology Workshop '93*, pages 266–271, Princeton, New Jersey.

Yarowsky, David. 1995. "Unsupervised word sense disambiguation rivaling supervised methods." In *Meeting of the Association for Computational Linguistics (ACL'95)*, pages 189–196.