Periods, Capitalized Words, etc.

Andrei Mikheev
University of Edinburgh *

In this paper we present an approach to tackle three important problems of text normalization: sentence boundary disambiguation, disambiguation of capitalized words when they are used in positions where capitalization is expected, and identification of abbreviations. We deliberately shaped our approach so that it does not largely rely on pre-compiled statistics. Our system does not try to resolve each ambiguous word occurrence individually. Instead, it scans the entire document for the contexts where the words in question are used unambiguously and this gives it grounds to resolve ambiguous contexts acting by analogy. The system is robust to domain shifts, new lexica and closely targeted to each document. A significant advantage of our approach is that it can be targeted to new domains completely automatically without human intervention. Although our approach requires two passes through a document, the simplicity of the underlying algorithms makes it reasonably fast. Despite its simplicity, the performance of our system was on the level with the previously highest reported results on the same test collections. Since our approach uses information derived from the global context it can be used complementary to the approaches based on the local context. We incorporated our system into a POS tagger and measured a significant cut in error rate for tagging the target categories. We also investigated the portability of our approach to other languages and obtained encouraging results on a corpus of news in Russian.

1 Introduction

Text cleaning and normalization is an important aspect in developing virtually any practical text processing application. Disambiguation of sentence boundaries, normalization of capitalized words as well as identification of abbreviations, however small in comparison to other tasks of text processing, are of primary importance at the developing of virtually any practical text processing application. These tasks are usually performed before actual "intelligent" text processing starts and errors made at such an early stage are very likely to induce more errors at the later stages of text processing and are therefore very dangerous.

Disambiguation of capitalized words in mixed-case texts hardly received much 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 (ambiguous) positions include the first word in a sentence, words in all-capitalized titles or table entries, a capitalized word after a colon or open quote, the first capitalized word in a list-entry, etc. 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

* 2 Buccleuch Place, Edinburgh EH8 9LW, UK

© 1994 Association for Computational Linguistics

just capitalized common words as in *"White elephants are ..."*.

The *disambiguation of capitalized words* in the ambiguous positions leads to the *identification of proper names* (or their derivatives) and in this paper we will use these two terms and the term *case normalization* interchangeably. Note that this task does not involve the classification of proper names into semantic categories (person, organization, location, etc.) which is the objective of the Named Entity Recognition task.

Many researchers observed that commonly used blind upper/lower case normalization does not necessarily help document retrieval. Church (Church, 1995), among other simple text normalization techniques, studied 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 are due to the fact that some capitalized words stand for proper names (such as *Continental* – the name of an airline) and some don't.

Proper names are the main concern of the Named Entity Recognition subtask (Chinchor, 1998) of Information Extraction. There the disambiguation of the first word of a sentence (and in other ambiguous positions) is one of the central problems – about 20% of Named Entities are used 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 multi-word 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 involve 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 normally proper names should be 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 unauthorized people. And in general, most of the tasks which involve text analysis will benefit from the robust disambiguation of capitalized words in mandatory positions into proper names and capitalized 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 text processing application – 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 signal a sentence boundary. However, there are cases when a period denotes a decimal point or is a part of an abbreviation and thus it does not signal a sentence 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 sentence boundary disambiguation problem can be found in (

The disambiguation of capitalized words and sentence boundaries presents a chicken and egg problem. If we know that a capitalized word which 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 the end-of-sentence marking is introduced by abbreviations: if we know that the word which precedes a period is *not* an abbreviation, then almost certainly this period denotes a sentence boundary. However, if this

word is an abbreviation, then it is not that easy to make a clear decision. This is also exacerbated by the fact that abbreviations do not form a closed set, i.e., one cannot list all possible abbreviations. It gets even worse – abbreviations can coincide with regular words, i.e., "in" can denote an abbreviation for "inches", "no" can denote an abbreviation for "number", "bus" can denote an abbreviation for "business", etc.

In this paper we present a method which tackles sentence boundaries, capitalized words, and abbreviations in a uniform way through the development of a *document-centered approach* (DCA). We investigate domain and genre portability and multi-linguality aspects of the new approach, evaluate it on well-known corpora and compare it to other state-of-the-art approaches. We also discuss the integration of the DCA into a part-of-speech (POS) tagging framework and thoroughly evaluate the combined system.

2 Performance Measure, Corpora for Evaluation and Intended Markup

A standard practice to measure the performance of a system for the class of tasks we are concerned with 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 it is intended to handle. Obviously, we want the system to handle all cases as accurately as possible. Sometimes, however, it is beneficiary to assign only cases where the system is confident enough leaving the rest unassigned and to be handled by other methods. In this case we also measure system's *coverage*

$coverage = \frac{all\_assigned\_by\_system}{all\_to\_be\_assigned}$

There are two corpora normally used for the evaluation on many tasks of text processing: 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 sub-corpora which belong to different genres and domains, ranging from news-wire texts and scientific papers, to fiction and transcribed speech. The Brown Corpus is rich with out-of-vocabulary (unknown) words, spelling errors and ungrammatical sentences with complex internal format. Altogether there are about 500 documents in the Brown Corpus with 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 are rich with abbreviations and proper names but in comparison to the Brown Corpus they are much shorter. Altogether there are about 2,500 documents in the WSJ Corpus with average length of about 500 word tokens.

Documents in the Brown Corpus and the WSJ are split into paragraphs and sentences. Sentences are tokenized and word-tokens are annotated with part-of-speech (POS) tags. Abbreviations are tokenized together with their trailing periods while fullstops and other sentence boundary punctuation is tokenized as separate tokens. This gives all necessary information for the evaluation of the performance on all our three tasks: the Sentence Boundary Disambiguation task, the Capitalized Word Disambiguation task and the Abbreviation Identification task. It is trivial to interpret POS tags associated with word-tokens as indicators whether a word-token stands for a proper name or for a common word. We considered proper nouns (`NNP`), plural proper nouns (`NNPS`) and proper adjectives[1] (`JJP`) to signal proper names, and all other categories were considered to signal common words or punctuation. Since proper adjectives are not included into

1 These are adjectives derived from proper nouns, e.g. "American", and they are normally written capitalized.

```
...<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 tokenization and markup. Tokens are represented as XML elements `W` where the attribute `C` holds part-of-speech 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'`.

the Penn Treebank tag-set, we had to identify and retag them ourselves with the help of a gazetteer. We also specially marked word-tokens in the positions where they were ambiguously capitalized i.e. sentence starting words, words in all-capitalized titles and sentences, words after opening brackets, quotes, etc.

There is also an issue of embedded sentence boundaries, i.e., when there is a sentence inside a sentence. This can be a quoted direct speech sub-sentence inside a sentence, this can be a sub-sentence embedded in brackets, etc. We consider closing punctuation of such sentences equal to closing punctuation of ordinary sentences. In all our experiments we treated embedded sentence boundaries in the same way as normal sentence boundaries.

To approach the evaluation of the SBD task we had to develop a new tokenization convention for periods. In the traditional 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 to always tokenize a period (and other punctuation) as a separate token when it is followed by a whitespace, 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 (`'*'`). Therefore, we had to split final periods from abbreviations, insert them as separate tokens and assign them with `'A'` or `'*'` tags according to whether the abbreviation was the last token in a sentence or not. Using the tags associated with periods in this new convention it is trivial to reconstruct whether the word preceding a period is an abbreviation or a regular word.

All these transformations were done automatically by applying a simple `Perl` script. However, we found quite a few infelicities in the original tokenization and tagging 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 part-of-speech information. An example of such markup is displayed on Figure 1.

3 Our Approach to Sentence Boundary Disambiguation

If we had at our disposal the entirely correct information on whether a word is an abbreviation and whether or not an ambiguously capitalized word is a proper name, we could apply a very simple set of rules to disambiguate sentence boundaries. Here is a simplified version of these rules listed in order in which they are applied:

- if a period follows a non-abbreviation 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 (`'*'`);

Table 1
Upper and Lower Bound estimates for our approach to the SBD task.

	Brown Corpus.			*WSJ Corpus.*		
	SBD	Cap. words	Abbrs.	SBD	Cap. words	Abbrs.
Upper Bound	0.01%	0.00%	0.00%	0.13%	0.00%	0.00%
Lower Bound	2.00%	7.40%	10.8%	4.10%	15.0%	9.6%
lookup proper names all correct abbrs.	1.20%	7.40%	0.00%	2.34%	15.0%	0.00%
all correct proper names guessed abbrs.	0.45%	0.00%	10.8%	1.96%	0.0%	9.6%

- 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 which is not a proper name it is sentence terminal and part of the abbreviation (`'*'`);

It is trivial 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, semicolons, etc., this rule-set also includes corresponding rules but since we are mostly concerned with the disambiguation of periods we decided not to list the entire SBD rule-set which we used in all our experiments.

The estimates from the Brown Corpus and the WSJ show that the application of this (entire) SBD rule-set together with the word-token information marked up in the corpora produces very accurate results (error rate less then 0.0001%) but it leaves unassigned the outcome of the case when 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 never end a sentence, whereas abbreviations of American states such as Mo., Cal., Ore., etc. when followed directly by a proper name most likely end sentence. Obviously encoding this kind of information into a system requires detailed analysis of the domain lexica, is not robust to unseen abbreviations and is labour intensive.

To make our method robust to unseen words we opted for a crude but simple solution. We estimated that if such ambiguous cases are always resolved as "not sentence boundary" (`'A'`), this produces the error rate of less than 3%. Estimates from the Brown Corpus and the WSJ showed that there are only 5-7% of such ambiguous cases among all potential sentence boundaries. This translates into a relatively small impact of the crude strategy on the overall error rate which was measured at 0.01% on the Brown Corpus and at 0.13% on the WSJ (as presented in the first row of Table 1). Although this over-simplistic strategy introduces a small penalty on the performance, we decided to use it because it is completely domain independent.

The described above SBD handling strategy is simple, robust and well performing, but it relies on the assumption that we have the entirely correct information on whether a word is an abbreviation and whether or not an ambiguously capitalized word is a proper name. The main difficulty is that when dealing with real-world texts, we have to solve the abbreviation and the proper name tasks ourselves, which is far from trivial. Thus the estimates which are based on the application of our simple method when using the 100% correctly disambiguated capitalized words and abbreviations can be considered as the upper bound.

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 a capitalized word in a mandatory position is a proper name or not is to apply a lexical lookup strategy (possibly enhanced with a morphological word guesser, e.g., (Mikheev, 1997)). Using this strategy all words which are not listed in the lexicon of common words usually are 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. This difference 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 all-capitalized titles which are also treated as ambiguously capitalized.

The simplest strategy for deciding whether a word which is followed by a period is an abbreviation or a regular word is to apply well-known heuristics which are 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 which could stand for an acronym or a proper name (e.g. BBN). 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 which classifies as abbreviations short words (with length less than five characters) which are followed by a period and then by a comma, lowercased word or a number. All other words are considered to be non-abbreviations.

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 which 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 leave about 9.5% of abbreviations in the text unrecognized. This brings the overall error rate of the abbreviation guessing heuristics to about 10%.

Using the information produced by the list lookup approach to proper name identification and the abbreviation guessing heuristics in conjunction with the SBD rule-set described above gave us 2.0% error rate on the Brown Corpus and 4.1% on the WSJ 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. The second row of Table 1 displays a summary for the lower bound results.

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 lexicon lookup strategy to proper name identification (third row 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 (fourth row of Table 1). In general each of the knowledge sources if produced perfect results managed to significantly reduce the lower-bound error rate for our SBD method. 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 is to use our simple SBD rule-set and to invest into the disambiguation of capitalized word and abbreviations.*

4 Document-Centered Approach to Proper Name and Abbreviation Handling

As we discussed above, the bad news (well, not really news) is that virtually any common word can potentially act as a proper name or part of a multi-word proper name. The same applies to abbreviations: there is no closed list of abbreviations and almost any short word can be used as an abbreviation. Fortunately, there is good news too: important words are typically used in a document more than once and in different contexts. Some of these contexts create very ambiguous situations but some don't. 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 known by the readers.

This is an observation which can be applied to a broader class of tasks. For example, people are often referred to by their surnames (e.g., "Black") but 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, they don't 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 name or a location. We simply want to distinguish whether a capitalized word in a particular occurrence acts as a proper name (or part of a multi-word proper name) or it is just a common word which is capitalized because it is used in a mandatory position. If we restrict our scope only 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 said later.."* 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 proper noun interpretation 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 has been used capitalized in an unambiguous context (not in a mandatory position), this increases the chances for this word to act as a proper name in mandatory positions in the same document. And conversely if a word is seen only lowercased, this increases the chances to treat it as a common word even when used capitalized in mandatory positions of the same document. This, of course, is only a general principle and will be further elaborated elsewhere in the paper.

The same logic applies to abbreviations. Although a short word which is followed by a period is a potential abbreviation, the same word when 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 lowercased word or a comma.

We call such style of processing a *document-centered approach* (DCA) since the information for the disambiguation of an individual word-token is derived from the entire document rather than from its immediate local context. Essentially this requires to perform unsupervised learning from each document under processing "on-the-fly". This differentiates DCA from the traditional corpus-based approach where learning is applied prior to the processing, can be performed with supervision and uses multiple documents of the training corpus.

5 Getting Abbreviations

The answer to the question whether or not a word token is an abbreviation or a regular word largely solves the sentence boundary problem. In the Brown Corpus 92% of po-

tential sentence boundaries come after a regular word. The WSJ Corpus is richer with abbreviations and only 83% of sentences there 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 in text. Since unidentified abbreviations are then treated as regular words, the overall error rate of the guessing heuristics was measured at about 10%. 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. To evaluate this approach, from a different corpus we collected a list of 270 abbreviations as explained in section 8. First we applied only the list lookup strategy to our two corpora i.e. only when a token was found in the list of 270 known abbreviations it was classified as an abbreviation. This gave us an unexpectedly high error rate of about 12% as displayed in the second row of Table 2 . When we investigated what went wrong 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 were not present in our abbreviation list. But such abbreviations are handled well by the abbreviation guessing heuristics.

When we applied the abbreviation list together with the abbreviation guessing heuristics (third row of Table 2) this gave us a very strong performance on the WSJ – 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 which is not too dissimilar to the WSJ and thus, this list contained many abbreviations found in the WSJ 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.

The surface guessing heuristics supplemented with a list of abbreviation are accurate but they still can miss some abbreviations. For instance, if an abbreviation like "sec." or "Okla." are followed by a capitalized word and are 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 which 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 document-centered approach (section 4) as described below.

Each word of length four or less which is followed by a period is treated as a potential abbreviation. To disambiguate potential abbreviations the DCA system looks over the entire document for the contexts where a potential abbreviation is used. First, the system can collect 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 this in fact is not an abbreviation but rather a regular word (non-abbreviation). To decide whether a potential abbreviation is in fact an abbreviation we look for the contexts where it is followed by a period and then by a lowercased 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 (non-abbreviation) and therefore it is a potential abbreviation. But if in the same document we detect a context *"lived in Hong Kong in 1993"* this indicates that "Kong" is normally written without a trailing period and hence is not an abbreviation. Having established that, we can apply this information to non-evident contexts and classify "Kong" as a regular word throughout the document. However, if we detect a context as *"Kong., said"* this indicates that in this document ""Kong" is normally written with a trailing period and hence is an abbreviation. This

Table 2
Error rate for different combinations of abbreviation identification methods

Corpus	*WSJ*	*Brown*
guessing heuristics (GH)	9.6%	10.8%
list lookup (LL)	12.6%	11.9%
GH + LL	1.2%	2.1%
GH + DCA	6.6%	8.9%
GH + DCA + LL	0.8%	1.2%

gives us grounds to classify "Kong" as an abbreviation in all its occurrences within the same document.

The DCA relies on the assumption that there is a consistency of writing within the same document. Different authors can write "Mr" or "Dr" with or without a trailing period but we assume that the same author (the author of a document) will write it in the same way consistently. There, however, can occur a situation when 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 e.g. "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 the unambiguous contexts but also their bigrams with the preceding word. Of course, as in 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. However, some cases can still be resolved on the basis of the bigrams e.g. the system will assign "C" to be an abbreviation in an ambiguous context *"John C. Research"* and it will assign "C" as a regular word (non-abbreviation) in an ambiguous context *"vitamin C. Research"*.

When neither unigrams no bigrams can help to resolve an ambiguous context for a potential abbreviation, the system decides in favor of the more frequent category for this potential abbreviation deduced from the current document. Thus if the word "In" was detected to act as a regular word (preposition) 5 times in the current document and 2 times as abbreviation (for the state Indiana), in the context where neither of the bigrams collected from the document can be applied, "In" is assigned as a regular word (non-abbreviation). The last resort strategy is to assign all non-resolved cases as non-abbreviations.

The fourth row of Table 2 shows the results when we applied the abbreviation guessing heuristics together with the DCA. On the WSJ the DCA reduced the error rate by about 30% however on the Brown Corpus its impact was somewhat smaller – about 18%. This can be explained by the fact that abbreviations in the WSJ have a much higher repetition rate which is very important for the DCA. We also applied the DCA together with the list lookup and the guessing heuristics. This produced the reduction in the error rate on abbreviation identification by about 30% in comparison with the "list and guessing heuristics" (GH+LL) configuration as can be seen in the last row of Table 2 .

6 Getting Capitalized Words

The second key task of our approach is the disambiguation of capitalized words which follow a potential sentence boundary punctuation sign. Apart from being an important task of text normalization, the information about whether or not a capitalized word which 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 – a document-centered approach which analyses on-the-fly the distribution of ambiguously capitalized words in the entire document. This is implemented as a cascade of simple strategies which was briefly described in (Mikheev, 1999).

6.1 The Sequence Strategy

The first DCA strategy for the disambiguation of ambiguously capitalized words is to explore sequences of words extracted from the contexts where the same words are used unambiguously with respect to their capitalization. We call it the *Sequence Strategy*. The rationale behind this is that if there is a phrase of two or more capitalized words starting from an unambiguous position (e.g. following a lowercased 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 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 lowercased word, a number, or a comma), the system memorize it and also memorizes its partial order sub-phrases *"Rocket Systems"*, *"Rocket Systems Co."*, *"Rocket Co."*, *"Systems Development"*, etc. If then in the same document *"Rocket Systems"* is found in a mandatory position (after a period), the system will assign the word "Rocket" as a proper noun because it is part of a multi-word proper name which was seen in the unambiguous context. A span of capitalized words can also internally include alpha-numerals, symbols and lower-cased words of length three or shorter. This allows the system to capture phrases like *"A & M"*, *"The Phantom of the Opera"*, etc. Partial orders from such phrases are generated in a similar way but with a restriction that every generated sub-phrase should start and end with a capitalized word.

The Sequence Strategy can also be applied to disambiguate common words. Since there the case information does not enable the system to determine boundaries of a phrase, only bigrams of the lowercased words with their following words are collected from the entire document. For instance, from a context *"continental suppliers of Mercury"* the system collects three bigrams: *"continental suppliers"*, *"suppliers of"* and *"of Mercury"*. Longer sequences and their partial orders are not generated because without syntactic information it is not possible in general to restrict the scope of dependencies in such sequences. Now, when the system encounters the phrase *"Continental suppliers"* in a mandatory position (e.g. after a period), it can use the information that in the bigram *"continental suppliers"* the word-token "continental" was written lowercased and therefore was unambiguously a common word. On this basis the system can assign ambiguously capitalized word-token "Continental" as a common word.

Dual application of the Sequence Strategy contributes to its robustness against potential capitalization errors in the document. The negative evidence (not-proper name) is used together with the positive evidence (proper name) and blocks the assignment when controversy is 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.

However, if in the same document the system detects an alternative evidence e.g. *"the President"* or *"the president"* where "the" is not part of the proper name, it then will block the assignment of "The" as a proper name in ambiguous usages of "The President" as unsafe.

Here are the results of applying the Sequence Strategy to our two corpora:

	Error Rate		Coverage	
	WSJ	*Brown*	*WSJ*	*Brown*
Proper Names	0.12%	0.97%	12.6%	8.82%
Common Words	0.28%	0.21%	17.68%	26.5%

The Sequence Strategy is extremely useful when dealing with names of organizations since many of them are multi-word phrases composed from common words. For instance, the words "Rocket" or "Insurance" can be used both as proper names and common words within the same document. The Sequence Strategy keeps contexts of the usages of such words within the same document 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%. Wrong assignments came from the erroneous capitalizations in the documents, but the low error rate confirms that this strategy is not over-sensitive to such errors. For tagging common words the Sequence Strategy was also very accurate (error rate less than 0.3%) covering 17% of ambiguously 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 there are in general longer than in the WSJ which allows for collecting more word bigrams from a document.

6.2 Frequent List Lookup Strategy

Frequent List Lookup Strategy applies the lookup of ambiguously capitalized words in two word-lists. The first list contains common words which are frequently found in the sentence-starting positions and the other list contains most frequent proper names. Both these lists are compiled completely automatically as explained in section 8. Note, however, that the Sequence Strategy is applied prior to the frequent list assignment and thus, a word listed in one of these lists will not necessarily be marked according to its list class. The list lookup assignment is applied only to the ambiguously capitalized words which have not been handled by the Sequence Strategy.

For instance, among such cases resolved by the Sequence Strategy was a multi-word proper name "To B. Super" where both "To" and "Super" were correctly identified as proper nouns and a multi-word proper name "The Update" where "The" was correctly identified as part of the magazine name. Both "To" and "The" are listed in the list of frequent common sentence-starting words but nevertheless the system handled them correctly. Here are the results of applying the Frequent List Lookup Strategy to our two corpora:

	Error Rate		Coverage	
	WSJ	*Brown*	*WSJ*	*Brown*
Proper Names	0.49%	0.16%	2.62%	6.54%
Common Words	0.21%	0.14%	64.62%	61.20%

The Frequent List Lookup Strategy produced an error rate of less than 0.5%. A few wrong assignments came from phrases like "Mr. A", "Mrs. Someone" and words in titles like *"I've got a Dog"* where "A", "Someone" and "I" were assigned as common words while 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 on the

Brown Corpus and under 3% on the WSJ, but it is extremely effective for common words – it covered over 60% of ambiguously capitalized common words.

6.3 Single Word Assignment

The Sequence Strategy is accurate, but it covers only 9-12% of potential proper names in ambiguous positions. The Frequent List Lookup Strategy is mostly effective for common words and covers only about 5% of proper names. To boost the coverage on the proper name category the system applies another DCA strategy which uses the information collected from the entire document. 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 lowercased, 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 lowercased (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 this time the Sequence Strategy and the Frequent List Strategy have been already applied, and all high frequency sentence-initial words were 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 multi-word proper name), it is still not safe to mark it as a proper name in ambiguous positions. The Single Word Assignment strategy handles well the so-called "unknown word" problem when some domain-specific lexica is missed 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.

One notable difficulty for the Single Word Assignment represent words which denote profession/title affiliations. These words when modifying a person name might require capitalization, e.g. *"Sheriff John Smith"*, but in the same document they can appear lower-cased, e.g. *"the sheriff"*. When the capitalized variant occurs only as sentence initial, and the Sequence Strategy cannot find repetitive bigrams, the Single Word strategy predicts that it stands for a common word. This, however, is an extremely difficult case even for human indexers – some writers tend to use certain professions such as Sheriff, Governor, Astronaut, as honorific affiliations and others tend to do otherwise.

A generally difficult case for the Single Word Assignment strategy is when a word is used both as a proper name and as a common word in the same document, and 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 by the Sequence Strategy which is applied prior to the Single Word Assignment and tackles such cases using n-grams of words. Here are the results of applying the Single Word Assignment to our two corpora:

	Error Rate		Coverage	
	WSJ	*Brown*	*WSJ*	*Brown*
Proper Names	3.18%	1.96%	18.77%	34.13%
Common Words	6.51%	2.87%	3.07%	4.78%

The Single Word Assignment 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 the error rate as high as 6.61% on the WSJ and the coverage below 5%.

6.4 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 implemented two simple heuristics:

- if a single capitalized word is 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 which 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 which 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 ambiguously capitalized proper names.

6.5 Tagging Proper Names: The Overall Performance

In general, the cascading application of the above described strategies achieved about 1% error rate but it left unclassified about 9% of ambiguously capitalized words in the Brown Corpus and 15% of such words in the WSJ. These results are summarized in the first row of the table below. When we concentrate on the impact of the individual strategies, we see that for the proper name category the most productive is the Single Word Assignment, then the "After Abbr." strategy, and then the Sequence Strategy. For common words the most productive is the Frequent List strategy and the Sequence Strategy.

Now we have to decide what to do with the remaining 10-15% of unassigned ambiguously capitalized word. To keep our system simple and domain independent we opted for the lexicon lookup strategy which we evaluated in section 3. This strategy, of course, is not very accurate but it is applied only to 10-15% of ambiguously capitalized words. The second row of the table below displays the results of applying the lexicon lookup strategy after the DCA. We see that the error rate went up by more than three times (2.9% on the Brown Corpus and 4.9% on the WSJ) but no unassigned ambiguously capitalized words are left in the text.

	Error Rate		Coverage	
	WSJ	*Brown*	*WSJ*	*Brown*
Cascading DCA	1.10%	0.76%	84.12%	91.76%
Cascading DCA and Lexicon Lookup	4.88%	2.83%	100%	100%

Table 3
Performance estimates on the SBD, Capitalized Word Disambiguation and Abbreviation Identification.

	Brown Corpus.			*WSJ Corpus.*		
	SBD	Cap. words	Abbrs.	SBD	Cap. words	Abbrs.
Upper Bound	0.01%	0.00%	0.00%	0.13%	0.00%	0.00%
Lower Bound	2.00%	7.40%	10.8%	4.10%	15.0%	9.6%
Best quoted	0.20%	3.15%	–	0.50%	4.72%	–
DCA	**0.28%**	**2.83%**	**0.8%**	**0.45%**	**4.88%**	**1.2%**
DCA (no abbr lex)	0.65%	2.89%	7.7%	1.41%	4.92%	6.4%
POS Tagger	0.25%	3.15%	1.2%	0.39%	4.72%	2.1%
POS Tagger + DCA	0.20%	1.87%	0.8%	0.31%	3.22%	1.2%

7 Putting It Altogether

After the abbreviations have been identified and capitalized words have been classified into proper names and common words, the system can carry out the assignments of the potential sentence boundaries using the rule-set described in section 3. This strategy is based on the observation that if we have at our disposal unambiguous (but not necessarily correct) information whether the word which precedes a period is an abbreviation or not and whether the word which follows this period is a proper name or not, then in mixed-case texts the only ambiguous outcome is generated by configurations where 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 brings some penalty on its performance. The first row of Table 3 summarizes the upper bound for our approach when we have the entirely correct information on the abbreviations and proper names as explained in section 3. There the erroneous assignments come only from the crude treatment of abbreviations which are followed by proper names.

The lower bound for our approach was estimated by applying the list lookup strategy for capitalized word disambiguation together with the abbreviation guessing heuristics to feed the SBD rule-set as described in section 3. The second row of Table 3 summarizes the lower bound results. Here we see a significant impact of the infelicities of the capitalized words and abbreviation handling on the performance of the SBD rule-set.

In the third row of Table 3 we summarized the highest known to us results for all our three tasks produced by automatic systems on the Brown Corpus and the WSJ. State-of-the-art machine-learning and rule-based SBD systems achieve the error rate of about 0.8-1.5% measured on the Brown Corpus and the WSJ. The best performance on the WSJ was achieved by a combination of the SATZ system (Palmer and Hearst, 1997) with the Alembic system (Aberdeen et al., 1995) – 0.5% error rate. The best performance on the Brown Corpus, 0.2% error rate, was reported by (Riley, 1989), who trained a decision tree classifier on a 25 million word corpus [2]. For the disambiguation of capitalized words the most wide-spread method is part-of-speech tagging. This achieves about 3% error rate on the Brown Corpus and 5% error rate on the WSJ as reported in (Mikheev, 2000). We are not aware of any studies devoted specifically to the abbreviation identification with comprehensive evaluation on either the Brown Corpus or the WSJ.

2 we don't consider here our own work on POS tagging of sentence boundaries (Mikheev, 2000) to which we will refer later.

In the fourth row of Table 3 we summarized our main results: the results obtained by the application of our simple SBD rule-set which uses the information provided by the DCA to capitalized word disambiguation applied together with the lexicon lookup (as described in section 6.5), and the abbreviation handling strategy which included the guessing heuristics, the DCA and a list of 270 abbreviation (as described in section 5). As can be seen the performance of this system is almost indistinguishable from the best previously quoted results. On the proper name disambiguation it achieved 2.83% error rate on the Brown Corpus and 4.88% error rate on the WSJ. On the SBD task it achieved a 0.28% error rate on the Brown Corpus and a 0.45% error rate on the WSJ. 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 about 0.3% increase in the error rate on the SBD task.

Since our DCA system 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 while very long documents possibly contain too many clues which cancel each other. The average length of the documents in the Brown Corpus is about 2,300 words and documents are distributed very densely around their mean, thus not much can be inferred about the dependency between the document length and the performance of the system apart form the fact that 2,000-3,000 word-long documents are handled well by the approach. In the WSJ corpus the average length of the document is about 500 words and therefore we can investigate the effect of short documents on the performance. We binned 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 bin as shown on Figure 2.

As can be seen on Figure 2 short documents of 50 words and 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 of 50 to 100 words long the error rate is still a bit higher than normal and for longer documents the error rate stabilized around 1.5 for the Capitalized Word task and 0.3 for the SBD task. The error rate on the documents of length 2,000 words and higher is almost identical to that registered on the Brown Corpus on the same document length. Thus here we can conclude that the proposed approach tends to be not very effective for the documents shorter than 50 words (one to three sentences) but it handles well the documents of up to 4,000 words long. Since our corpora did not contain documents significantly longer than that, we could not estimate whether/when the performance of our method would significantly deteriorate on longer documents. We also evaluated the performance of the system on different subcorpora of the Brown Corpus: the most difficult subdomains proved to be scientific, spoken language transcripts and journalistic, while fiction was the easiest genere for the system.

To test the adaptability of our approach to a completely new domain we applied our system in a configuration where it was *not* equipped with the list of 270 abbreviation since this list is the only domain-sensitive resource in our system. The results for this configuration are summarized in the fifth row of Table 3 . The increase of 5-7% in the error rate on the abbreviation handling introduced about twofold increase in the error rate on the SBD task on the Brown Corpus (a 0.65% error rate) and about threefold increase on the WSJ (1.41%). But these results are still in the range of the performance of the majority of currently used sentence splitters.

To test our hypothesis that DCA can be used complementary to a local context approach we combined our main configuration (evaluated in the fourth row of Table 3) with a POS tagger. Unlike other POS taggers, this POS tagger was also trained to disambiguate sentence boundaries as described in (Mikheev, 2000). The performance of

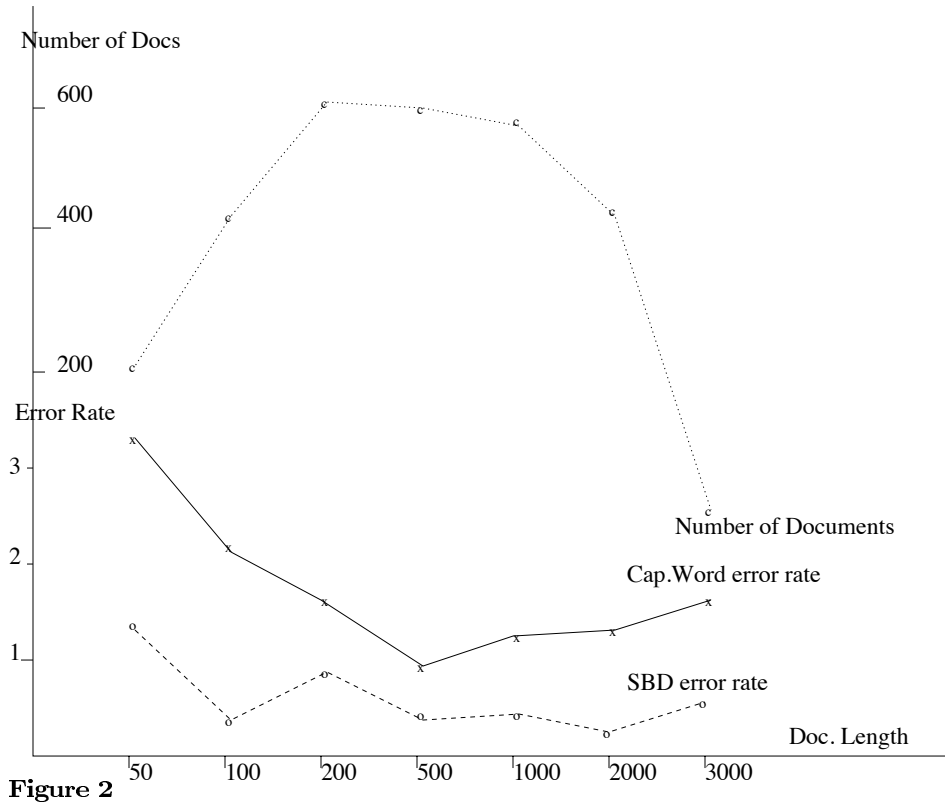

Figure 2
Distribution of the error rate and the number of documents across the document length in the WSJ corpus.

the tagger alone (sixth row of Table 3) is very close to that of the DCA. We felt that the two approaches are complementary to each other, since they use different types of information: global distribution and local context. Thus, the hybrid system can bring at least two advantages. First, unassigned by the document-centered approach 10-15% of the ambiguously capitalized words can be assigned using a standard POS tagging method based on the local syntactic context rather than inaccurate lexical lookup approach. Second, the local context can correct some of the errors made by the document-centered approach.

To implement this hybrid approach we incorporated the DCA system to the POS tagging model by simple linear interpolation. The last row of Table 3 displays the results of the application of the hybrid system. We see an improvement on proper name recognition by about 1.5%: the overall error rate of 1.87% on the Brown Corpus and the overall error rate 3.22% on the WSJ. This in its 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, which corresponds to about 20% cut in the error rate in comparison to the DCA and the POS tagging approaches alone.

8 Building Resources

There are only four word lists which are used by our method. Each list is a collection of words which belong to a single type. Since we have four lists we have four types:

- common word

- common word which is a frequent sentence starter
- frequent proper name
- abbreviation

All these lists are acquired completely automatically from raw (unlabeled) texts. For the development of these lists we used a text collection of about 300,000 words derived from The New York Times (NYT) Corpus, which 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 paper was initially designed to be part of our Named Entity recognition system (Mikheev, Grover, and Moens, 1998) developed for MUC'7. In the development of these four lists we used the NYT text collection not annotated in any way. Although the size of 300,000 words can be seen as large, the fact that these words are not annotated in any way and that a similar corpus can be easily collected from the Internet makes this step virtually labour-free.

The first list our method relies on is a *list of common words*. This list is used for the lexicon lookup strategy which is applied to the ambiguously capitalized words which hasn't been covered by the DCA strategies as explained in section 6.5. This list includes common words for a given language but no supplementary information such as part-of-speech or morphological information is required to be present in this list. There is a variety of such lists for many languages already available e.g. (Burnage, 1990), and usually words in such lists are also supplemented with morphological and part-of-speech information. However, we don't need to rely on the preexisting resources – a list of common words can be easily automatically obtained from a raw (unannotated in any way) text collection by simply collecting and counting lowercased words in it. To smooth for potential spelling and capitalization errors we included in the list of common words only words which occurred lowercased at least three times in the corpus. The list of common words which we developed from the NYT collection included about 15,000 words.

The second list that our method uses is a *list of common words which are most frequently seen in sentence starting positions*. We use this list as a default strategy to assign sentence starting words as described in section 6.2. This list can also be obtained completely automatically. Remember, that the lexicon lookup strategy for capitalized word disambiguation (Table 1) performs with an 8-15% error rate. We used the described above list of common words over the NYT corpus to tag capitalized words in mandatory positions as common words and as proper names. Of course, this tagging was far from perfect but it was good enough for our purposes. We included in the frequent starters list only 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 can guess the most frequent sentence starting word was "The". This list also included some adverbs such as "However", "Suddenly", "Once", some prepositions such as "In", "To", "By" and even a few verbs – "Let", "Have", "Do", etc.

The third list we develop is a *list of single-word proper names* which 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, thus we would like to include this word in the list of proper names. We use this list as a default strategy to assign sentence starting words as described in section 6.2. We included in the *proper name list* 200 words which were most frequently used in the NYT corpus as single capitalized words in unambiguous positions and which at the same time were present in the list of common words. For instance, the word "The" can be frequently seen capitalized in the unambiguous positions but it is always followed by another capitalized word, and we don't count it as a candidate. On the other hand the word "China" is often seen capitalized in

unambiguous positions and is not preceded or followed by other capitalized words, thus we can include it into the proper name list. Applying this strategy we produced a list of the 200 most frequent proper names.

The fourth list is a *list of known abbreviations*. To induce this list our strategy is to apply the abbreviation guessing heuristics described in section 5 to a corpus and then extract most frequent abbreviations. We applied this strategy to out NYT text collection and extracted 270 most frequent abbreviation. This list included honorific abbreviations (Mr, Dr), corporate designators (Ltd, Co), month name abbreviations (Jan, Feb), abbreviations of American states (Ala, Cal), measure unit abbreviations (ft, kg), etc. Although we described these abbreviations in groups, we did not encode this information in the list – the only information this list provides is that a word is a known abbreviation.

It is important to note that since the compilation of all four lists does not require data pre-annotated 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 corpus. From 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 recreated for a new domain. Although this process is completely automatic and does not require any human labour apart from collecting an unannotated sample corpus sometimes this is not possible. For instance, in the situation when the system is used to handle documents from unknown origin. However, our evaluation showed (section 7) that the system can still produce reasonable performance even when not equipped with the abbreviation list.

9 Further Experiments

9.1 The Cache Extension

One of the features of the method advocated in this paper is that the system performs on-line unsupervised learning for each document, then it applies this information during the second pass through the document (actual processing) and then it "forgets" what it has learned before handling another document. The main reason for not carrying the information, which has been learned from one document to process another document, is that in general we don't know that this new document comes from the same corpus and thus, the regularities which have been learned 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 the documents contain enough disambiguation clues. However, as we showed in section 7, when handling short documents of one to three sentences long, quite often there are not enough disambiguation clues within the document itself which leads to an inferior performance.

To improve the performance on short documents we introduced a special caching module which propagates some information learned from previous 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. However, word sequences are much more stable and can be propagated across documents. We decided to accumulate in our cache all multi-word proper names and lowercased word bigrams which are induced by the Sequence Strategy (section 6.1). These word sequences are used by the Sequence Strategy exactly as 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 which are induced by the abbreviation handling module as explained in section 5. These bigrams are used together with the bigrams induced on-the-fly in the same way.

This strategy proved to be quite useful: it covered another 2% of unresolved cases

with the error rate of less than 1%.

9.2 Handling Russian News

To test the multi-linguality of our approach we applied it to a corpus of BBC news in Russian. We collected this corpus from the Internet[3] over a period of 30 days. These gave us a corpus of 300 short documents - one or two paragraphs each. We automatically built the supporting resources from 364,000 Russian corpus of European Corpus Initiative (ECI) as explained in section 8.

Since, unlike English, Russian is a highly inflected language, we had to deal with the case normalization issue. Before using our DCA method, we applied a Russian morphological processor (Mikheev and Liubushkina, 1995) to converted each word in the text to its main form – singular nominative case for nouns and adjectives, infinitive for verbs, etc. For words which could be normalized to several main forms (polysemy), when secondary forms of different words coincide, we retained all the main forms. Since the documents in our BBC news corpus were rather short, we applied the cache module as described in section 9.1. 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 their disambiguation of capitalized words in mandatory positions easier. The overall performance reached a 0.1% error rate on sentence boundaries and a 1.8% error rate on ambiguously capitalized words with the coverage on both tasks at 100%.

10 Related Research

10.1 Research in Non-Local Context

The use of non-local context and dynamic adaptation have been studied in *language modeling for speech recognition*. Kuhn and de Mori (1998) proposed a cache model which works as a kind of short-term memory by which the probability of the 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 but unlike the cache model, it uses a multi-pass strategy.

Clarkson and Robinson (1997) developed a way of incorporating the cache model with standard n-grams, 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 part-of-speech tagger. Instead of decaying non-local information we opted for not propagating it from one document for processing of another. For more general application of the DCA we will probably need to experiment with the information decay.

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 to view proper names 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 paper also uses non-local discourse context and does not heavily rely on pre-existing word lists. It has been applied not only to the classification of proper names (Mikheev, Grover, and Moens, 1998) but also to their identification.

3 `http://news.bbc.co.uk/hi/russian/world/default.htm`

Gale, Church and Yarowsky (1992) showed that words strongly tend to exhibit only one sense in a document or discourse. This observation is also implicitly used in our *document-centered approach* especially for the identification of abbreviations. In the capitalized word disambiguation we, however, use this assumption with a caution and first apply strategies which rely not just on single words but on words together with their local contexts (n-grams).

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

10.2 Research in Text Preprocessing

There exist two large classes of SBD systems: rule based and machine learning. The rule based systems use manually built rules which 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 labour consuming enterprise. For instance, the Alembic workbench (Aberdeen et al., 1995) contains a sentence splitting module which employs over 100 regular-expression rules written in Flex. Another well acknowledged shortcoming of rule-base 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 quickly re-trainable for a new corpus, domain or even for another language. Thus, the second class of SBD systems employs machine learning techniques such as decision tree classifiers (Riley, 1989), maximum entropy modeling (Reynar and Ratnaparkhi, 1997), neural networks (Palmer and Hearst, 1997), etc. 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 training data annotation phase. Machine learning systems treat the SBD task as a classification problem, using features such as word spelling, capitalization, suffix, word class, etc., found in the local context of potential sentence terminating punctuation.

Our approach to SBD is closer to machine learning methods than rule-based ones. The main difference between the existing methods and our approach is that we decided to tackle the SBD task through the disambiguation of the preceding and the following a period words and then feed this information into a very simple SBD rule-set. In contrast to that the standard practice in building SBD software is to disambiguate configurations of a period with its local context directly, preserving the ambiguity of the words on the left and on the right.

Disambiguation of capitalized words is usually handled by part-of-speech (POS) tagging. POS taggers treat capitalized words in the same way as other categories i.e. by accounting for the immediate syntactic context and using the estimates collected from a training corpus. However, as Church (1988) rightly pointed out, "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 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 ambiguously capitalized words has always been tightly linked to the classification of proper names into semantic classes such as person name, location, company name, etc. Named entity recognition systems usually use sets of complex hand-crafted rules which employ a gazetteer and a local context (Krupka and Hausman, 1998). In some systems such dependencies are supervisedly 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 it 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 not easily portable.

Both POS taggers and named-entity recognizers are normally built using the local context paradigm. In contrast we opted for a method which relies on the entire distribution of a word in a document rather than on its local context. Although it is possible to train some classes of POS taggers unsupervisedly the majority of taggers are trained from labeled data. Named Entity recognition systems are usually hand-crafted or trained from labeled data. Our approach does not require supervised training since it performs learning "on-the-fly" unsupervisedly for each particular document.

Not much information has been published on abbreviation identification. One of the better known approaches is described in (Grefenstette and Tapanainen, 1994) who suggested first to extract abbreviations from a corpus using abbreviation guessing heuristics akin to those described in section 5 and then to reuse these abbreviations in further processing. This is similar to our treatment of abbreviation handling but our strategy has been applied on the document rather than corpus level. The main reason for restricting the abbreviation discovery to a single document is that this does not presuppose access to a corpus where the current document is essentially similar to other documents.

11 Discussion

In this paper we presented an approach which tackles three problems: sentence boundary disambiguation, disambiguation of capitalized words when they are used in positions where capitalization is expected and identification of abbreviations. All these tasks are important tasks of text normalization, which is a necessary phase in almost all text processing activities.

In general, 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 to a potential sentence boundary punctuation sign;
- to disambiguate capitalized words and abbreviations we use global information distributed across the entire document rather than the immediate local context;
- our approach does not require manual rule construction or data annotation for training, it uses very simple resources which can be acquired completely automatically from raw (unlabeled) corpus;
- by applying on-line unsupervised learning rather than relying on resources smoothed across the entire document collection our approach is closely targeted to each document under processing.

We deliberately shaped our approach so that it does not largely rely on pre-compiled statistics. This is because the most interesting events are inherently infrequent and, hence, are difficult to collect reliable statistics for, and at the same time pre-compiled statistics would be smoothed across multiple documents rather than targeted to a specific document. Our system does not try to resolve each ambiguous word occurrence individually. Instead, it scans the entire document for the contexts where the words in question are used unambiguously and this gives it grounds to resolve ambiguous contexts acting by analogy. The system is robust to domain shifts, new lexica and closely targeted to each document.

A significant advantage of our approach is that it can be targeted to new domains completely automatically without human intervention. The four word lists which our system uses for its operation can be acquired from a raw corpus and do no require any human annotation. Although, some SBD systems can be trained from relatively small sets of labeled examples, their performance in this case is somewhat lower than their optimal performance. For instance, (Palmer and Hearst, 1997) report that the SATZ system (decision tree) was also trained on a labeled sample set of about 800 periods. This is a relatively small training set which can be manually marked in a few hours time. But the error rate of the decision tree trained on this small sample (1.5%) was about 30% 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 raw (unannotated in any way) corpus of texts. Although, it needs such corpus to be relatively large (a few hundred thousand words) this is usually not a problem since when the system is targeted to a new domain such corpus is usually already available at no extra cost. For instance, our system was "trained" on 300,000 word corpus from New York Times without any human labor, and when applied to the same test data as the SATZ system, its error rate on sentence boundary was 0.44%[4]. So in our approach there is no tradeoff between the amount of human labor and the performance. This not only makes retargeting of such system easier, but also allows it to be operational in completely autonomous way: it only needs to be pointed to texts from a new domain and then it can retarget itself automatically.

Although our approach requires two passes through a document, the simplicity of the underlying algorithms makes it reasonably fast. It processes about 3,000 words per second on Pentium II 400 MHz. This includes the disambiguation of sentence boundaries, identification of abbreviations and disambiguation of capitalized words. This is comparable to the speed of other preprocessing systems[5]. The training speed is about 10% lower than the operational speed because apart from applying the system to the training corpus this also involves collecting, thresholding and sorting of the word lists. Training on the 300,000-word NYT corpus took the system about 2 minutes.

Despite its simplicity, the performance of our system 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 almost as good as the lowest quoted before (Riley, 1989) (0.28% vs. 0.20% error rate). On the WSJ 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 where it is difficult to justify the error rate less than 2% because even human annotators have a disagreement rate of about

4 This result was obtained on subsections 2 to 6 of the WSJ.

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

3%, sentence boundaries are much more unambiguous (the disagreement of about 1 in 5,000). This shows that the error rate of 1 in 200 (0.5%) is still far from reaching the upper bound. On the other hand, one error in 200 periods means that there is one error in every two documents in Brown Corpus and one error in every four documents in the WSJ. On the proper name disambiguation task the system performed very close to a POS tagger (3–5% error rate) as shown in table 7.

With all its strong points there are a number of restrictions for the proposed approach. First, in its present form it is suitable to be used for processing of reasonably "well-behaved" texts which consistently use capitalization (mixed-case) and do not contain too much noisy data. Thus, for instance, we don't expect our system to perform well on single-cased texts (e.g. written in all capital letters) or on OCR generated texts. We also investigated in section 7 that very short documents of one to three sentences long also present difficulty to our approach. This is where robust syntactic systems like SATZ (Palmer and Hearst, 1997) or a POS tagger reported in (Mikheev, 2000), which do not heavily rely on word capitalization, have their advantage.

Our DCA system uses information derived from the global context and thus can be used complementary to the approaches based on the local context. When we incorporated our DCA system into a POS tagger (section 7) we measured a 30-35% error rate cut on proper name identification in comparison to the DCA and the POS tagging approaches alone. This in its 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, which corresponds to about 20% cut in the error rate.

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 for the majority of European languages, since they share the same principles of capitalization and word abbreviation. An obvious exception, though, is German where capitalization is used not only for proper name and sentence start identification. This, however, does not mean that the document-centered approach in general is not suitable for preprocessing of German texts – it just needs to use different clues.

Initially the DCA system was developed as part of our named entity recognition system (Mikheev, Grover, and Moens, 1998) which participated in the 7th Message Understanding Conference. There the ability to identify proper names with high accuracy proved to be instrumental for the entire system to achieve a very high performance. Since then the text normalization module has been used in several other systems and its ability to be easily adopted to new domains enabled rapid development of text analysis systems in medical, legal and law enforcement domains.

Acknowledgments

The work reported in this paper was supported in part by grant GR/L21952 (Text Tokenization Tool) from the Engineering and Physical Sciences Research Council, UK, and also it benefited from the ongoing efforts in building domain independent text processing software at Infogistics Ltd.

References

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

Baldwin, B., C. Doran, J. Reynar, M. Niv, B. Srinivas, and M. Wasson. 1997. Eagle: An extensible architecture for general linguistic engineering. In *Proceedings of Computer-Assisted Information Searching on internet (RIAO '97)*, Montreal, June.

Bikel, D., S. Miller, R. Schwartz, and R. 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. 1995. Transformation-based error-driven learning and natural language parsing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

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

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

Church, K. 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, K. 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, P. and A.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.

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

Gale, W., K. Church, and D. Yarowsky. 1992. One sense per discourse. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pages 233–237.

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

K. Seymore, S. Chen and R. Rosenfeld. 1988. Nonlinear interpolation of topic models for language model adaptation. In *Proceedings of The 5th International Conference on Spoken Language Processsing (ICSLP'98)*, Sydney.

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

Mani, I. and T.R. MacMillan. 1995. Identifying unknown proper names in newswire text. In B. Boguraev and J. Pustejovsky, editors, *Corpus Processing for Lexical Acquisition*. MIT Press, 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, A. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405–423.

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

Mikheev, A. 2000. Tagging sentence boundaries. In *Proceedings of the 1st Meeting of the North American Chapter of the Computational Linguistics (NAACL'2000)*, pages 264–271, Seatlle, Washington. Morgan Kaufmann.

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

Mikheev, A., C. Grover, and M. 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, VA*, Fairfax, VA. Morgan Kaufmann.

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

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

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

16–19, Washington, D.C. Morgan Kaufmann.

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