Using SRX standard for sentence segmentation in LanguageTool

Marcin Miłkowski, Jarosław Lipski

Polish Academy of Sciences,
Institute of Philosophy and Sociology
ul. Nowy Świat 72, 00-330 Warszawa, Poland
mmilkows@ifispan.waw.pl, loomchild@rootnode.net

Abstract

In this paper, we evaluate using the SRX (Segmentation Rules eXchange) standard for specifying sentence segmentation rules created for a proofreading tool called LanguageTool. As proofreading tools are quite sensitive to segmentation errors, the underlying segmentation mechanisms must be sufficiently reliable. Even though SRX allows only regular expressions as a means for specifying sentence breaks and exceptions to those breaks, our evaluation shows that it is sufficient for the task, both in terms of the performance of the algorithm used and correctness of results. Moreover, it offers interoperability with different tools, which in turn allows maintaining consistent segmentation in a whole language-processing pipeline. Our rules are available on an open-source license (LGPL), which also helped in receiving valuable community input from our users.

Keywords: segmentation, standards, computer-aided translation, grammar-checking

1. Introduction

Most natural language processing tasks such as syntactic parsing, information extraction, machine translation, bilingual sentence alignment, document summarization or grammar checking, require that the input text first be segmented into sentences before the higher-level processing can be performed. At first, sentence segmentation seems to be a straightforward process, at least in languages in which certain punctuation marks, such as a period, an ellipsis, a question mark and an exclamation mark, are used to denote sentence boundaries. Nevertheless, in many cases the period occurs in an abbreviation, as a decimal point, in a date, or in many other places and does not signal a sentence break.[1] Furthermore, the period can be used at the same time as part of the abbreviation and as a sentence-end mark when the abbreviation is the last word in the sentence. Additionally, some abbreviations may have exactly the same spelling as ordinary words at the end of a sentence, and disambiguation of such cases of homophony may be non-trivial without deeper semantic analysis. Therefore, sentence segmentation can pose a serious challenge for a computer.

In this paper we focus only on sentence segmentation of the written text that includes punctuation. We measure the performance of our system using two classical metrics: recall and precision. Precision is the ratio of correctly recognized sentence breaks to total sentence breaks found by the system. Recall is the ratio of correctly recognized sentence breaks to total actual sentence breaks in the text.

Sentence segmentation algorithms can be divided into two major classes: rule-based and statistical (Mikheev, 2003). Rule-based algorithms require manually prepared set of rules, often specified in terms of regular expressions. In these systems, the difficulty lies in manually creating a good set of rules, which is time consuming. Statistical algorithms employ machine training techniques and treat segmentation as a classification problem. The majority of these techniques require a labelled corpus for supervised training (Palmer, Hearst, 1997), while some of them can perform unsupervised learning on a raw corpus (Schmid, 2000). However, statistical algorithms do not offer interoperability with different tools as there is no standard way to exchange machine-learned information for segmentation, and that limits the usage of such methods to standalone applications that are not used in a complex linguistic pipeline.

Sentence segmentation has applications that are not equally sensitive to segmentation quality. Let us present some of the examples, ordered by the ascending level of sensitivity to correctness of segmentation.

• **Information retrieval from a corpus**. If search criteria do not depend on sentence boundaries, segmentation has no effect. Even if they do, segmentation quality has minor impact on search quality and it is generally better to not segment than to segment in unclear cases.[2]

• **Bilingual sentence alignment.** If segments are too long, correct alignment may become impossible because sentences are atomic units of this process. If the segments are too short, it can be harder to find the correct alignment because there are more degrees of freedom. Generally, it is better to split too often than too seldom as the alignment process can fix some segmentation errors by mapping many sentences in one language to one sentence in another language.

• **Computer-Aided Translation (CAT) systems with Translation Memory**. Segmentation errors are not that significant as long as segmentation rules used for the input text are the same as those used for creation of translation memory; otherwise, there will be fewer matches or they will tend to be less accurate. Moreover, in heterogeneous environments where translators use different tools, the bilingual documents returned to a central repository cannot be fully leveraged for future translations by including them in the Translation Memory if translators did not use the same segmentation.

[1] According to (Gale and Church, 1991) in the Brown corpus 90% of the periods denote sentence break, whereas in the Wall Street Journal only 53% do.

[2] Sometimes it is possible to limit search to one sentence for example by using “within s” notation in Poliqarp system (Przepiórkowski, 2004).

• **Statistical machine translation.** Corpus segmentation quality is moderately important. If segmentation produces too long sentences, for example by classifying abbreviation as not ending the sentence, the word alignment will be less accurate, as the complexity of this task is directly related to sentence length. If segmentation produces too short sentences, for example by not recognizing an abbreviation, some word alignments can be left unnoticed. However, as long as there are no major word order differences in source and target sentences it is better to segment than not to segment in unclear cases.

• **Proofreading tools.** Segmentation quality becomes very important as soon as errors cause false positives for proofreading tools in cases where linguistic error detection depends on sentence breaks. In the proofreading tool under consideration in this paper, LanguageTool[3], there are several such cases. First, there is a generic rule for many languages that detects a lower case character at the beginning of the sentence, and it can create a false positive in case where an abbreviation was not detected as such and the sentence was falsely segmented. Additionally, for Polish, there are rules that detect two clauses that are not linked with a punctuation mark nor with a conjunction. In this case, not detecting a sentence end can result in a false positive; it is usually the case when an ordinary word is classified as an abbreviation.

In this paper, we focus on a rule-based approach to sentence segmentation of written text in Polish and English. Obviously both languages use punctuation for signalling sentence breaks. The rules have been created for LanguageTool and replaced our previous segmentation algorithm. As the results obtained were promising – both in terms of speed and accuracy – the rules have been created by developers of other language modules. At the time of writing of this paper, there are also SRX rules for Dutch, Romanian, Russian, Slovak, and Icelandic.

2. SRX standard

SRX (Segmentation Rules eXchange) standard (SRX, 2008) defines an XML vocabulary for describing the rules used for breaking a text document into segments. In other words, it is a formal notation that can be used by various tools to specify segmentation rules, in particular sentence-level segmentation. It was created as an addition to TMX (Translation Memory eXchange) standard (TMX, 2005) to enable interoperability between different Translation Memory Management Systems. This is achieved by exchanging segmentation rules along with the translation memory, so input text can be segmented the same way in different tools that use the standard. As a side effect, SRX provides separation between the segmentation algorithm and segmentation rules, which enables non-programmers to improve the rules without modifying the algorithm. This is why we could receive direct input from translators who used our rules.

SRX file is divided into two parts. The first part, represented by <languagerules> element, specifies the segmentation rules. The second part, represented by <maprules>, specifies which segmentation rules are applied to which language.

To segment a text after a full stop, an exclamation mark and a question mark but not segment it after “Mr.” in English and “prof.” in Polish, language rules can be defined as follows:

```
<languagerule
  languagerulename="Polish">
<rule break="no">
<beforebreak>\s[Pp]rof\.</beforebreak>
<afterbreak>\s</afterbreak>
</rule>
</languagerule>
<languagerule
  languagerulename="English">
<rule break="no">
<beforebreak>\sMr\.</beforebreak>
<afterbreak>\s</afterbreak>
</rule>
</languagerule>
<languagerule
  languagerulename="Default">
<rule break="yes">
<beforebreak>[\.\?!]+</beforebreak>
<afterbreak>\s+[A-Z]</afterbreak>
</rule>
<rule break="yes">
<afterbreak>\n</afterbreak>
</rule>
</languagerule>
```

Each language rule consists of one or more segmentation rules, and each segmentation rule consists of two regular expressions that must match the text before a break position and after the break position for the rule to be applied. Each segmentation rule has an attribute saying if it is a break rule (break="yes") or exception rule (break="no"). Regular expressions used in the rules must conform to a subset of ICU[4] regular expressions as defined in SRX specification. Our implementation uses the Java standard library regular expression engine which supports almost everything required by SRX with a few minor omissions[5]. The order of segmentation rules is important as they are matched from first to last.

Continuing the previous example mapping rules can be defined as follows:

```
<languagemap
  languagepattern="(PL|pl).*"
  languagerulename="Polish"/>
<languagemap
  languagepattern="(EN|en).*"
```

[3] LanguageTool is an open-source rule-based proofreading tool for English, German, Polish, Dutch, and other languages that integrates with OpenOffice.org suite. For more information see project homepage: www.languagetool.org.

[4] ICU is a set of C/C++ and Java open-source libraries providing Unicode and globalization support for software applications. For more information see project homepage: http://site.icu-project.org.

[5] In Java, there is no special behaviour of \b character within a set and missing support for \Uhhhhhhhh and \x{hhhh} constructs. As far as we know, currently all tools supporting SRX standard are implemented in Java, so their behaviour is the same as in our implementation.

```
  languagerulename="English"/>
<languagemap
  languagepattern=".*"
  languagerulename="Default"/>
```

Each language mapping consists of a language regular expression to which language code is matched and a language rule corresponding to it. Language codes are defined in (RFC4646). As SRX supports the concept of cascading, for Polish text identified by "pl" language code both Polish language rule and Default language rule will be merged and applied together. Language patterns are matched and aggregated in appearance order.

To avoid confusion how to segment using these rules, the following algorithm pseudocode is included in specification's implementation notes:

```
for each inter-character text position
    for each rule in list
        if current rule matches
        inter-character position
            if rule specifies break
                break text here
            end if
            exit for
        end if
    next
next
```

To make the process more efficient, implementation we used in Segment library[6] differs from the above in that it first matches break rules and searches for exception rules only in potential break positions. In the worst case, our algorithm performs the same as the above, but in a typical scenario, when there are few break rules and many exception rules, it performs much faster - observed speed increase was more than tenfold for real-world scenario.

SRX standard is more complicated than outlined in this section in that it allows formatting of the input text to be preserved. In current implementation, we ignore the formatting and treat input as plain text.

3. Disambiguation strategies

One of the difficulties with specifying segmentation rules in SRX format is that it is limited to regular expressions over surface strings. It is not possible to refer to part-of-speech (POS) information, not only because segmentation precedes POS tagging in most cases but also due to the fact that SRX does not include any provisions for additional linguistic information.

To maintain interoperability with other SRX-compliant tools one should limit rules to what is offered by the standard. In other words, though one could add additional segmentation in later phases of linguistic processing to enhance precision of the segmentation, any such addition means that the SRX file will no longer be useful to recreate the same sentence boundaries in linguistic data. It is however not a critical limitation of the standard, as experimental results show that regular expressions allow sufficiently high precision and recall to be maintained. Moreover, as SRX is offered for software packages that lack such resources as POS taggers, especially in computer-aided translation, any such addition to the SRX standard would make it much harder to implement in a wide variety of tools.

In writing SRX rules, one faces two kinds of ambiguity: (1) homophonic ambiguity between words and abbreviations that end with a punctuation character; (2) ambiguity between abbreviations that end the sentence and those that do not. The Table 1 lists some such homophones in Polish (see also Rudolf 2004 and Mazur 2005).

Abbreviation	Expansion of abbreviation	Ordinary word
farm.	farmakologia	farma (plural genitive)
gen.	generał	gen
im.	imienia	oni (plural dative)
jap.	japoński	japa (plural genitive)
klas.	klasyczny	klasa (plural genitive)
kop.	kopiejka	kopać (imperative)
lic.	licencjat	lico (plural genitive)
marsz.	marszałek	marsz
min.	minister, minuta, minimum	mina (plural genitive)
muz.	muzyczny	muza (plural genitive)
par.	paragraf	para (plural genitive)
por.	porównaj	por
pot.	potocznie	pot
sen.	senator, senior	sen
tłum.	tłumaczył	tłum
ul.	ulica	ul
ust.	ustęp	usta (plural genitive)
żart.	żartobliwie	żart
żeń.	żeński	żenić się (imperative)

Table 1. *Homophones in abbreviations for Polish.*

Additionally, abbreviations may be created *ad hoc* in a given text. This means that the class of abbreviations in a language is always open. Yet, as (Mazur 1996) observes, in many languages abbreviations do differ in their surface form from other words; for example, in Polish, abbreviations may contain no vowels in contradistinction to ordinary words. By using this information, one may devise a simple heuristic rule that any lower case string of alphabetic consonants is an abbreviation. Another heuristic is that a sequence of single characters with dots (such as "U. S. A.") is an unbreakable abbreviation (it can, however, end the sentence).

Polish spelling rules require that a dot be placed only after such abbreviations that end with the same character as the word being abbreviated. This knowledge is

[6] Segment is an open-source tool used to split text into segments according to rules stored in SRX file. For more information see the project homepage: http://sourceforge.net/projects/segment.

unavailable during segmentation for newly made abbreviations, which again makes the second ambiguity problem harder.

There are several linguistic strategies that can be used to tackle the ambiguities. First of all, it could seem that the easiest strategy to deal with dot-ended abbreviations would be to create a long list of all of them. The strategy is however flawed if this results with more ambiguities between words and abbreviations. For example, the abbreviation "klas." (klasyczny [classical]) is a homophone of "klas" (plural genitive of klasa [class]), yet the second word is much more frequently used in Polish than the abbreviation. In this case, better results could be achieved by removing "klas." from the abbreviation list. In other words, it is better to limit the abbreviation list to the ones which are actually often used and do not tend to be ambiguous. And ambiguity can be resolved using heuristics based on the following observations.

There are several categories of abbreviations. Some abbreviations may occur either in the middle or at the end of the sentence. In such a case, in many languages the character case may be used as a source of information: if there is no upper case character followed by a lower case character after the dot ("...abc. Abc..."), probably the sentence ends with an abbreviation. Or, in a worse case, the next word is a German noun or a proper noun.

Not all words and abbreviations may occur at the sentence end. For example, personal titles in English, such as "Dr." or "Mr.", are most often followed by proper names.

Some abbreviations precede only certain kinds of strings. For example, a Polish abbreviation "ust." (meaning "ustęp" [clause]) is used in legal contexts only before Arabic numbers larger than 0 and smaller than 100 (the upper bound is a heuristic, though in principle one could have a legal contract that contains an article with a hundred of internal clauses, or more). In contrast, the word "ust" (genitive of "usta" [lips]) is hardly followed by a number. (In general, numbers rarely start sentences, if not used in tables and similar contexts.)

Some abbreviations occur frequently in the same surface patterns as groups. For example, abbreviations of personal titles are written in the same order ("prof. dr", and not "dr prof."). In general, as abbreviations are characteristic for more official registers, they also belong to more formulaic language. One may therefore easily get collocation information from corpora.

Some abbreviations that should not end with a dot, according to official spelling rules, are actually frequently misspelled with a dot. Detecting such a blunder in LanguageTool would be impossible if they were not included in segmentation rules, so one should take special care to list them among abbreviations that never end the sentence (if they actually never do). In Polish, this includes abbreviations such as "wg", "dr" or "nr".

A special category of abbreviations are initials, or upper case alphabetic characters with a dot (or one upper case and one lower case character, for example "St" in Polish for "Stanisław"). They always precede other initials or upper case words. Yet, as single-letter abbreviations rarely end a sentence, this can be used to enhance segmentation quality. For example, in Polish, only three one-character abbreviations seem to occur at the sentence end: "r." (rok [year]), "s." (strona [page]), "n." (następny [next]), and they are all lower case[7].

Preceding punctuation marks can also be used to disambiguate between abbreviation kinds: if an abbreviation is used in parentheses, the dot cannot be used to end the sentence (for example, "(gr.)"). Similarly, "tłum." (tłumaczył [translated by]) is usually preceded with a comma, and its homophone "tłum" (crowd) usually not.

Additionally, white space can indicate whether the dot is used as a sentence break. Dots used inside dates in German and Polish ("12.10.1995") are not followed by any white space. The same is true for dots in URLs ("www.languagetool.org").

All these strategies cannot guarantee correct results in all possible cases. For example, a potential sentence-breaking abbreviation may be followed by a proper noun. It is hardly possible to create a rule that would deal with such a case properly, as the information required to disambiguate is of the semantic or even pragmatic nature. (Rudolf, 2000) cites such a hard example: "Jednym z najtrwalszych dzieł młodej królowej było odnowienie przez nią w 1387 r. Akademii Krakowskiej, założonej jeszcze w 1364 r. przez Kazimierza Wielkiego, która jednak upadła po jego śmierci." [One of the most durable achievements of the young queen was to renew the Academy of Cracow in 1387, which was already established in 1364 by Kasimir the Great but fell into decay soon after his death].

Our SRX rules cut the sentence after "1387 r.", which is hardly correct for semantic reasons. We think, however, that it is inevitable in machine-created segmentation.

On a limited scale, one might be tempted to use a lexicon with part-of-speech information to create a regular expression that would allow to make a limited reference to parts-of-speech just by enumerating frequent words with a standard regular-expression disjunction (word1|word2|word3). In LanguageTool, the POS tagger used later in the processing is implemented as a finite-state machine, so in principle, one could translate any piece of information from the tagger to a regular expression. Yet, performance limitations of regular expression processing (large disjunctions are known to introduce serious performance penalty) make this translation less feasible a solution. For that reason, we did not try to implement a "poor-man's POS tagger" in SRX this way.

4. Results for English and Polish

Initial SRX rules we used were based on rules created for LanguageTool. These rules were also built using regular expressions but hard-coded in Java and therefore harder to maintain than a SRX file. The old algorithm was also more than three times slower than the current, generic one implemented in Segment library[8]. These rules were

[7] Note that in technical language, one may use initials for enumeration or in technical terms, such as "part A" or "kinase B" (see below for information on problems with GENIA corpus). In those cases, this heuristic rule breaks.

[8] Segment was able to split about 2500 Polish sentences (330000 characters) per second on an old, single core 1GHz processor computer which proved sufficient for us.

subsequently enriched with additional heuristics. As a test corpus, we used the GENIA molecular biology corpus with manually created and verified segmentation marks (Kim et al. 2003). For the GENIA corpus, we assumed that the original manual segmentation is always correct. The recall rate was 99.10%, and precision 94.34%. Manual review of results has shown that in the GENIA corpus, there are high numbers of technical terms such as "kinase B" that tend to end sentences, and our rules assumed that single letters can never end a sentence. It must be remembered, however, that the GENIA corpus is a specialized corpus and for literary text the precision may be actually higher.

For Polish, however, there is no corpus with manually verified sentence end markers. To evaluate the rules, we used the Frequency Dictionary Corpus[9]. However, the corpus contains errors that can heavily influence the performance of rules. Although the original corpus was proofread, it was also automatically converted to encode Polish characters (early versions of the computer representation of the corpus could not encode them properly), and due to the automatic conversion and transliteration of old data, all Polish diacritical characters at the sentence beginnings are now lower case. Also, all characters following the question and exclamation marks are lower case, whether they end the sentence or not. All these errors were artificially introduced by using a Perl script (Nazarczuk, 1997). For that reason, we used two versions of the corpus: raw and cleaned (using LanguageTool standard uppercase-sentence start rule and adding a missing dot to complete sentences). Yet, the differences seem to be not too significant: the recall is 97%, and precision 99.7548% for the raw corpus; the edited version has the same recall but precision is slightly lower (99.7525%). As the original corpus was not evaluated manually and contains segmentation errors, the lower precision rate may be actually misleading in the latter case. All in all, we think that the differences are insignificant.

As the above results show, by using regular expressions over surface tokens, one can achieve quality which is sufficient for practical applications. SRX standard simplifies maintaining and implementing the rules, though the overall quality of segmentation clearly depends on the linguistic knowledge embedded in the heuristics used.

5. Conclusions

SRX standard has its limitations but offers interoperability with other tools. Interoperability means also that the rules of segmentation can be easily enhanced and extended by users. This was the case of the current set of rules; CAT tools such as Swordfish, though already containing quite extensive rule sets, benefit from using better segmentation. We received valuable input from the community of translators who added some of the missing but common abbreviations and are interested in extending the linguistic coverage of our segmentation file.

In LanguageTool, correct segmentation is crucial for avoiding false positives that would greatly reduce usability of the grammar checker. We could add special sentence-breaking rules on a higher level of linguistic analysis that allows using such mechanisms as part-of-speech tagging and unification for disambiguating particularly hard cases. Though strict interoperability in terms of keeping the consistence of segmentation is not so important for us, as LanguageTool output is not used to create further linguistic resources, we think that community feedback is so valuable that we do not want our extensions to become incompatible with other tools. Moreover, for Polish, as it seems, we already achieved such high precision that the economic law of diminishing returns applies: the work spent at making the rules perfect cannot result in dramatic improvement. There is more room for improvement in the English rules and this is where we will turn our attention.

[9]We have used the proofread version, available at http://www.mimuw.edu.pl/polszczyzna/pl196x/.

References

Gale, W.A. Church, K.W. (1991) A Program for Aligning Sentences in Bilingual Corpora. Meeting of the Association for Computational Linguistics.

ICU. http://site.icu-project.org.

Kim, J., T. Ohta, Y. Teteisi and J. Tsujii. (2003). GENIA corpus - a semantically annotated corpus for bio-textmining. Bioinformatics. 19 (suppl.1) (pp. i180-i182).

LanguageTool. http://www.languagetool.org.

Mazur, P.P. (2005). Text segmentation in Polish. 5th International Conference on Intelligent Systems Design and Applications (pp. 43-48).

Mikheev, A. (2003). Text segmentation. The Oxford handbook of Computational Linguistics (pp. 201-218). Oxford University Press.

Nazarczuk, M. (1997). Wstępne przygotowanie korpusu Słownika frekwencyjnego polszczyzny współczesnej do dystrybucji na CD-ROM. Praca magisterska napisana pod kierunkiem dra hab. Janusza S. Bienia. Warszawa.

Palmer, D. D. Hearst, M. A. (1997). Adaptive multilingual sentence boundary disambiguation. Computational Linguistics 23(2) (pp. 241-269).

Przepiórkowski, A. (2004). The IPI PAN Corpus: Preliminary version. IPI PAN, Warszawa.

RFC4646. Tags for Identifying Languages. Internet Engineering Task Force. http://www.ietf.org/rfc/rfc4646.txt.

Rudolf, M. (2004). Metody automatycznej analizy korpusu tekstów polskich. Uniwersytet Warszawski, Warszawa.

Schmid, H. (2000). Unsupervised learning of period disambiguation for tokenization. Internal Report. University of Stuttgart.

Segment. http://sourceforge.net/projects/segment.

SRX (2008). SRX 2.0 Specification. The Localization Industry Standards Association. http://www.lisa.org/fileadmin/standards/srx20.html.

TMX (2005). TMX 1.4b Specification. The Localization Industry Standards Association. http://www.lisa.org/tmx/tmx.htm.