A Trainable Tokenizer, solution for multilingual texts and compound expression tokenization

Oana Frunza
School of Information Technology and Engineering
University of Ottawa Ottawa, ON, Canada, K1N 6N5
ofrunza@site.uottawa.ca

Abstract

Tokenization is one of the initial steps done for almost any text processing task. It is not particularly recognized as a challenging task for English monolingual systems but it rapidly increases in complexity for systems that apply it for different languages. This article proposes a supervised learning approach to perform the tokenization task. The method presented in this article is based on character transitions representation, a representation that allows compound expressions to be recognized as a single token. Compound tokens are identified independent of the character that creates the expression. The method automatically learns tokenization rules from a pre-tokenized corpus. The results obtained using the trainable system show that for Romanian and English a statistical significant improvement is obtained over a baseline system that tokenizes texts on every non-alphanumeric character.

1 Introduction

Tokenization is the process of isolating word-like units from a text (Grefenstette and Tapanainen 1994); the process of mapping sentences from character strings into strings of words (Guo 1997).

Even though we may think that tokenization is not very important, it can be a step that influences the final results of a more complex task. For example McNamee and Mayfield (2004) show that the results for an information retrieval system based on N-gram tokenization for several European languages is influenced by the length of the N-grams. They also show that an average N-gram word tokenization representation outperforms a space-based word tokenization representation because raw words work relatively worse with morphologically complex languages.

High-level tasks (e.g., machine translation, named entity recognition) can influence the way tokenization is done. Grefenstette and Tapanainen (1994) explain how the Brown corpus is tokenized differently from the Susanne corpus, both being English corpora but designed for different tasks.

The method that is proposed in this article differs than the traditional methods by its supervised learning approach and by the fact that it can solve the non-alphanumeric character attachment (e.g. hyphen in hyphenated words), a problem for languages like Romanian and French, languages that tend to have a large number of compound expressions.

For Bio-Informatics texts, tokenization is an important step that can influence the results for tasks like information extraction, gene discovery, gene interaction, etc. Drug names, gene names, are most of the time compound expressions that can contain eater spaces eater hyphens eater a different non-alphanumeric character that defines the multiword expression (e.g. G-protein-coupled Receptor Kinase, Glycogen SYnthase). A system that can automatically learn and apply tokenization rules for all non-alphanumeric characters can play an important role for high-level tasks performed on medical domain texts.

2 Related Work

The traditional approach to perform the tokenization task uses hand-crafted rules with regular expressions and/or finite state automata (NLTK (Loper and Bird 2002)). MtSeg, a system developed by P. di Cristo[1] is a rule-based tokenization system that can be tuned for various languages. The system was enhanced with the corresponding resources for several Western European languages including Romanian by Tufis *et al.* (1998). A second approach for tokenization is based on sequence labeling algorithms such as Hidden Markov Models (HMMs). In Gil *et al*. (2002) tokenization and part-of-speech tagging are done in one step using an HMM algorithm.

This paper addresses the tokenization task as a stand-alone problem and not performing it in combination with other tasks. The method proposed in this article does not use human expertise or any type of dictionaries, the only source of information required is a pre-tokenized corpus; the language knowledge is hypothesized to be encoded in a training corpus. A comparison of the proposed trainable method with a traditional hand crafted rule-based system, BaLIE[2], is performed.

[1] http://aune.lpl.univ-aix.fr/projects/multext/MtSeg/
[2] http://balie.sourceforge.net/

3 Problems in Tokenization

Ambiguity: An ambiguity occurs for every unit of text that can play multiple syntactic roles within a sentence or within different languages. For example in the Romanian and French big numbers (e.g. thousands, millions, etc.) contain dot as a mark, while for English comma is used.

Round-up: The round up problem appears when multi–word expressions need to be recognized as single tokens. Having lists of compound words can help but sometimes they are not available for all languages. Moreover, languages are under a continuous process of change new words and new expressions are added to a language all the time

Hyphenation: Hyphenation is problematic when it serves the purpose of end-of-line hyphen and most importantly for languages that allow multi-word expressions to be created with the hyphen. (e.g vowel elision – "ont-ils", in French and "dintr-un", in Romanian). The hyphen attachment is an important problem in the Romanian language.

4 Trainable Tokenizer

This article proposes a supervised learning approach for the tokenization task. The method requires a training corpus where lexical units (tokens) are identified. Usually part-of-speech tagged corpora are well suited for this task since they already have the lexical units divided.

The method presented in this article is evaluated on a Romanian and English corpus, Orwell's novel *1984*, from the MulText[3] project. Romanian was chosen as a language of study because is not a very resourceful language and also because the tokenization task differs in the way it is done in English.

4.1 Instances and Features

To create a training corpus to be used by the ML algorithm, both the raw text version (textual, readable version) and the pre-tokenized versions (one token per line) of the novel are used.

The method uses two types of textual units to create the training corpus:

1. **Sequence of alphanumeric characters.** The sequences represent consecutive alphanumeric characters from the raw text. A sequence of alphabetic characters is considered as an atomic unit since it is highly intuitive that for segmented languages there is not a split inside such a sequence in order to create different tokens.

2. **Non-alphanumeric characters.** This unit contains a single character. It can be either a punctuation character or another special character that is not alphanumeric, but not a space. Consecutive non-alphanumeric characters are not merged in a unit since they are often part of different tokens.

3. Besides type 1 and type 2 text units, **space is also identified** as a special character. The space is not considered a non-alphanumeric unit of type 2 since it has the property of being a good token divider but it is never considered a token by itself.

An instance is created on each transition between textual unit of type 1 or 2 and for each space in the raw text. Our goal is to learn to classify transitions as a **Split** or a **Not-Split** decision.

The features used are:

IsASpace (boolean): *true* if the character that creates the current transition is a space, *false* if not.

PrecedingSegment (nominal): the string values of the preceding text unit (type 1 or type 2) of the character that creates the transition.

FollowingSegment (nominal): the string value of the following unit of text (type 1 or type 2) of the character that creates the transition.

Class: (Split or Not-Split). The class of a given instance can be **Split** – the current transition (the character that creates the transition) is a token delimitation, a token ends and another token start, **Not-Split** – the current transition is not delimitating two tokens.

Figure 1: Sample instances created on each transition for a Romanian text.

The tokenized corpus contains the following tokens:
{"stia", "ca", ",", "intr-un fel", "sau", "altul", ","}

Each segment becomes an instance:

Instance i1: true, stia, ca, Split
Instance i2: false, ca, ',', Split
Instance i3: true, ',', intr, Split
Instance i4: false, intr, -, Not-Split(etc.)

If we look at Instance i4, we can see that this text representation and the fact that we consider every non-alphanumeric character as a transition can capture the hyphen attachment that is needed to be solved for a correct tokenization.

4.2 Datasets

The referenced Romanian pre-tokenized novel that is used in the experiments has 117,866 tokens. About 5.5% percent of the tokens are compound (multi-word expressions that contain space or hyphen). From 3,478 hyphen appearances in the corpus 1,788 times it was part of a multi-word expression and 1,301 times it was attached to a token. 126,185 instances are created

[3] http://nl.ijs.si/ME/V3/doc/

(117,865 with the class Split and 8,321 with the class Not-Split).
The English version of the novel contains 113,400 tokens with 116,308 instances created. 1% of the total numbers are compound tokens. The Split class contains 113,399 instances and the Not-Split class 2,909. The class imbalance for the English version of the novel is more than double as for the Romanian data set. This is explained by the fact that the Romanian language contains more locutions and hyphenated words than English. This observation supports the fact that specific rules need to be applied for each language.
The class imbalance for our data set is 1:14 for the Romanian version and 1:38 for the English version. No rebalance was done because the natural distribution found in the corpus was meant to be preserved. Even though the imbalance is high, the results show that the method is able to significantly outperform a baseline that will always predict the majority class, in this case the Split class.

5 Evaluation and Results

The evaluation was done using a 10-fold cross-validation technique for both versions of the novel. ZeroR classifier was chosen to be used as a baseline (always predicts the majority class, the Split class). Naïve Bayes and AdaBoost are additional classification algorithms used from the Weka[4] Tool.

Classifier	Accuracy_RO	Accuracy_ENG
ZeroR (Baseline)	93.40%	97.40%
AdaBoost	93.49%	99.57%
Naïve Bayes	**95.50%**	**99.70%**

Table 1. Accuray results obtained with the trainable tokenizer.

Some experiments using simple tokenization rules were also performed. For this experiments **CTS** – number of correctly recognized tokens divided by the total number of tokens in the pre-tokenized corpus is reported.

Splitting Characters	CTS RO	CTS ENG
Space	67.6%	75.8%
" ",,".","!",",",":",";","?","/","(",")"	94.4%	98.2%
All non-alpha/numeric	94.3%	98.4%
Balie[5] System	94.4%	99.1%

Table 2. Results for simple tokenization rules.

[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://balie.sourceforge.net

5.1 Discussion of Results

For both Romanian and English corpus the improvements that were obtained are 0.95 statistically significant on the t-test even when using highly imbalanced data sets. For the Romanian language the majority of errors were done for the Not-Split class on multi-word expressions, compound adverbs and conjunctions. Examples of errors that were made by the method on the Not-Split class are presented in Figure 2.

Figure 2. Errors on the Not-Split class for Romanian.
*Text1: **din moment** ce inceputul continea*
Instance i1: true, din,moment, Not-Split

*Text2: Pe de**-o** parte*
Instance i2: false, -, o, Not-Split

For the English version they were done on the Split class on punctuation marks (e.g. “.”, “?”) and some made on apostrophe. Some examples of the errors on the split class are presented in the following figure:

Figure 3. Errors on the Split class for English.
Instance i1: true,”.”, Charrington, Split
Instance i2: false, conspirator, ? , Split
Instance i3: false, artist, """, Split

The obtained results show that for both corpora a trainable tokenization is significantly better than a hand-crafted rule-based tokenization. From Machine Learning (ML) point of view, the tokenization task is associated here with a classification task. The possible classes used in the classification task are Split and Not-Split and are used by the method to determine if a non-alphanumeric character is or not a token delimitation. The system also outperformed several hand-crafted rule based systems and was also shown to be able to determine non-alphanumeric token attachment, a problem that was not especially addressed before.
Although tokenization is thought to be an *easy task* this work is motivated by the need to adapt tokenization to languages and sometimes to tasks.

6 Conclusions and Future Work

Although tokenization is thought to be an *easy task* this work is motivated by the need to adapt tokenization to languages and sometimes even to tasks.
The evaluation step, for the method proposed in this paper was done on corpora from two different languages: Romanian and English. The system outperformed several hand-crafted rule based systems and statistically outperformed a base-line system. Besides the encouraging results, the method also showed to be able to determine non-alphanumeric token attachment, a problem that was not especially addressed before.

Appling the method to other languages, choosing different data representations ─ looking at more context features and using syntactic/semantic relations features are just few of the main ideas that are part of the future work plans.

7 Acknowledgements

The work presented in this paper was done while I was also affiliated to the IIT Group of the National Research Council of Canada.

I would like to thank David Nadeau from the IIT Group, National Research Council of Canada for all the help and without whom this work would not have been possible.

8 Translations

dintr-un / **dintre** = from / among;
nici un = none;
stia ca, intr-un fel sau altul = he knew that in a way or another;
din moment ce inceputul continea = because the beginning contained;

9 References

Grefenstette, G. and Tapanainen, P. (1994) What is a Word, what is a Sentence? Problems of Tokenization. *International Conference on Computational Lexicography*, Budapest, 79-87.

Gil, J. G., Alonso, M. A. and Ferro, M. V. (2002) A Common Solution for Tokenization and Part-of-Speech Tagging. *Proceedings of the 5th International Conference on Text, Speech and Dialogue.*

Guo J. (1997) Critical Tokenization and its Properties, *Computational Linguistic*, 23(4), pp.569-596.

Loper, E. and Bird, S. (2002) NLTK: The Natural Language Toolkit. *ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*. New Jersey, US.

McNamee, P., and Mayfield J. Character N-gram Tokenization for European Language Retrieval. Information Retrieval, 7(1-2):73-97, 2004.

Tufis, D., Ide, N. Erjavec, T (1998) Standardized Specifications, Development and Assessment of Large Morpho-Lexical Resources for Six Central and Eastern European Languages. *In Proceedings of LREC*, Granada, Spain, pp. 233-240.