The Stakes of multilinguality : Multilingual Text Tokenization in Natural Language Diagnosis*

Emmanuel Giguet
GREYC — CNRS URA 1526 — Université de Caen
Esplanade de la Paix
14032 Caen cedex — France
e-mail: Emmanuel.Giguet@info.unicaen.fr

In this article, we address a problem occurring in any multilingual treatments. This problem is interesting since it involves what we can call the stakes of multilinguality.

The major problem occuring in multilingual text treatments is that at any moment the language can change and linguistic treatments must manage this event to prevent failure. A way of preventing failure is to use a function called *Natural Language Diagnosis* or *Categorization According to Language*. The idea is to tag each part of the input with the name of its language.

The problem we are going to give a solution to, is that at any moment we never know the current language since it is the goal of our tool to discover it. It is impossible to choose the right monolingual tokenization rule database. So, how can we do a blind but efficient tokenization of a multilingual text ?

First, it seems important to explain the importance of multilinguality for our research. Then, we will illustrate step by step our claims in describing the way we solve our problem.

After introducing the linguistic knowledge used for Natural Language Diagnosis, we will present the advantages of studying the problem of tokenization in a multilingual framework and we will give an elegant answer to both Monolingual and Multilingual Text Tokenization.

Then, we will present a way of modifying the standard monolingual tokenization approach to take into account multilingual requirements.

1 The Stakes of Multilinguality

First, it seems important for us to explain the stakes of working in a multilingual context. These stakes are both linguistic and computational.

Dealing with multilinguality is fundamental for our research. It is a whole side of our working method. It is a way to structure and improve our knowledge on natural languages. In fact, multilingual treatments bring to make common linguistic features appear and to isolate language specific ones. We can say that multilinguality implies generalization.

Back to monolingual reflections, we notice that a detour by multilingual considerations is also a way of improving the knowledge on one particular language. We claim that these improvements could not have been obtained without this detour.

The computational improvements are mainly conceptual ones. Knowledge engineering take advantages of the separation between common and language specific features.

We are going to illustrate in concrete terms these claims. The framework states in the Natural Language Diagnosis and the problem is to find a good way of tokenizing an input.

2 The Framework: Natural Language Diagnosis

An answer to Natural Language Diagnosis[1], also called *Categorization According to Language*, has been given in (Giguet, 1995b; Giguet, 1995a). The aim of such a tool is to tag sentences with the name of their language.

This research is both parallel and complementary to some standardization efforts which are done for language engineering in large projects such as Text Encoding Initiative[2], Eagles[3] and Multext[4]. It is parallel since it is very useful for a NLP system to

*This Paper is published in the Proceedings of the 4th Pacific Rim International Conference on Artificial Intelligence Workshop "Future issues for Multilingual Text Processing", Cairns, Australia, August 27.

[1] The papers and a demo are available on the web: http://www.info.unicaen.fr/~giguet

[2] http://www-tei.uic.edu/orgs/tei

[3] http://www.ilc.pi.cnr.it/EAGLES/home.html

[4] http://www.lpl.univ-aix.fr/projects/multext

know the language which is processed. It is complementary since quantities of texts are already available and will never be standardized. Furthermore we don't believe that in the future authors will waste their time tagging by hand their own documents to help a NLP system understanding what they write.

One problem we have to solve was the following : "To process an input we have to tokenize it. But among the known languages, which tokenization rule database is to be used since the Natural Language Diagnosis is not done yet ?"

Before answering to this question, we briefly describe the linguistic properties used for our Natural Language Diagnosis tool.

2.1 Linguistic Features used for Natural Language Diagnosis

The process is based on the study of natural properties of languages. The power of resolution is based on the combination of several methods. For each of them, we try to find linguistic justifications. Efficiency is not the goal, it has to be the consequence of a good linguistic analysis.

The main method is the search of *grammatical words*. In fact, they are proper to each language and are in a whole different from one language to another. Moreover, they are short, not numerous and we can easily build an exhaustive list. They represent about 50% of the words of a sentence. With this method and tested for 4 Western-Europe natural languages discrimination, the system achieve perfect categorization for sentences of more than 9 words.

The goal is now to categorize short sentences. To do this, we have to caracterize non grammatical words because short sentences don't have enough grammatical words to allow total discrimination. The other methods used are the alphabet and the word endings. The *alphabet* is useful thanks to characters with diacritics. The *word endings* is a compromise for non grammatical words caracterization. In fact, it is very difficult to get a representative corpus of one language, so relying on other informations is quite risky. When we add these two methods, the system achieve perfect categorization for sentences of more than 7 words and a very high discrimination rate for small sentences.

Among the different languages there can be interferences since a grammatical words or a character may exist in several languages (e.g French:*on* vs. English:*on*) but it is always the convergence of several clues which allows the categorization.

Since research on Natural Language Diagnosis has already been reported in (Giguet, 1995b; Giguet, 1995a), interested readers can get more precisions about this work in these articles or by getting in touch with me.

3 Multilingual Text Tokenization for Natural Language Diagnosis

The problem is that we don't know which monolingual tokenization rule database is to be used because the Natural Language Diagnosis is not done yet.

3.1 The first experiment

In the first version of our system, a french basic tokenizer was used to tokenize multilingual texts. The results were good since a lot of tokenization rules are common to all the targetted languages but many tokens from other languages were erroneous. In fact, the way of splitting tokens is sometimes different from one language to another even if it seems at first to be the same. Let's take a look at the elision. An elision often means that a voyell has disappeared. Depending on the language, several ways of splitting the elision exist. Here are some examples :

- In french, an elision is done between a pronoun, a determiner or a conjunction and the following word if it starts by a voyell. the elided voyell is always the last voyell of the first word (e.g *le avion* $\rightarrow$ *l'avion* $\rightarrow$ *l'* + *avion*). Another phenomenon is the elision in popular language. In this case, there is only one token (e.g *petit* $\rightarrow$ *p'tit* $\rightarrow$ *p'tit*).

- In english, with verb contractions, the elided voyell is the first of the second word (e.g *they are* $\rightarrow$ *they're* $\rightarrow$ *they* + *'re*). With negation contraction, the tokenization is different since the elision is done inside the negation (e.g *does not* $\rightarrow$ *doesn't* $\rightarrow$ *does* + *n't*).

- In italian, we note same ways of splitting than in french (e.g *della arte* $\rightarrow$ *dell'arte* $\rightarrow$ *dell'* + *arte*).

In this first experiment, the consequences of using a french basic tokenizer were very bad. We had to modify the grammatical word databases of the Natural Language Diagnosis to take into account the french tokenizater's errors. For instance, *they're* was split *they'* + *re* and *doesn't* was split *doesn'* + *t* because in french it is the easiest way of splitting the elision. So, we had to consider *they'* and doesn' as english grammatical words which was not acceptable in a linguistic point of view, even if the results were improved.

Splitting correctly tokens is fundamental for our system. In fact, we claimed in §2.1 that the more grammatical words appear in a sentence, the best Natural Language Diagnosis is. So it is important for our system to split contractions or inverted pronouns since this process often make grammatical words appear.

To settle this problem, we first thought about updating the french rules and adding rules to the french tokenizer to get a multilingual tokenizer but it was not a good solution. In fact, even if we have to update the french rules which were too basic, adding rules would not only generate an unmaintainable set of multilingual rules but this set would also be redundant with any monolingual tokenization rule database (i.e the english rules we should add to the multilingual tokenizer would be redundant to the rules included in a traditionnal english tokenizer).

From this point, we could not continue our reseach without studying precisely what meant tokenization in a multilingual framework.

3.2 Studying Tokenization in a Multilingual Framework

Studying problems of tokenization in five Western-Europe natural languages[5], we found that most of the rules were common to all these languages. These rules process tokens bounded by explicit separators like spaces and punctuation.

The language specific rules split tokens where no explicit boundaries can be located. For instance, one would like to split verb contractions in english *that's, couldn't* (or the Anglo-Saxon genitive of nouns), determiner, pronoun and conjunction contractions in french *l'envie, j'aime, qu'elle*, inverted pronouns in french *donne-le, veux-tu* and determiner contractions in italian *dell'arte.*

In a conceptual point of view, this multilingual analysis leads us to consider:

- one shared database for every languages,
- one language specific rule database for each particular language.

In a linguistic point of view, this study clarify the (monolingual) tokenization mechanism. In a computational point of view, instead of having one unstructured and quiet unreadable rule database, the two kinds of rules are physically divided into two databases that have to be merged to tokenize one particular language. For instance, to tokenize english, the shared database and the english specific rule database have to be merged.

[5] french, english, german, spanish and italian

To tokenize multilingual texts, we experiment the merge of the monolingual tokenizers. Merging the tokenizers means combining their tokenization rules.

3.3 Combining monolingual tokenization rule databases

We start the study with the background reported in section §3.2. There, we saw that many tokenization rules are common to every languages and that it was interesting to put them in a shared database. To tokenize a multilingual text, this set of rules just have to be executed once for all the targetted languages.

The problem of combining the language specific rules still remains. We choose to experiment the following idea. To tokenize a monolingual text, we have to merge the shared database and the language corresponding to the language of the text. To tokenize a multilingual text, a solution can simply consist in merging the shared database and all the language specific rule databases of the targetted languages, and then in applying the rules to the input.

Merging the shared database and a language specific rule database seems to be easy since the two sets of rules match different inputs. The first one matches patterns with explicit separators whereas the second one matches patterns with no explicit separators.

The problem one could fear states in the language specific rules. Are there interferences between the specific rules of the different languages? Until now, we never see such interferences but we have got some explanations.

The specific tokenization rules of a language process tokens with no explicit boundaries. These rules are often written to make grammatical words appear. With respect to the linguistic features described in §2.1, the grammatical words are not numerous. So, there are a few percent of them which can be involved in agglutination via a dash or an elision. Moreover, from one language to another, the grammatical words are in a whole different. According to us, an interference could states between two rules involving a grammatical word and an elision or a dash and the two rules don not define the cut at the same place. An example can be found in french where the string *rendez-vous* can be whether the noun *rendez-vous* or the conjugated verb *rendre* following by the inverted pronoun *vous.*

The problem is now to check if the monolingual standard approach to tokenization suits these multilingual requirements and allows us to realize such a model. We will see that it does not and that we show how to modify it.

4 Problems of tokenization

In the former section, we saw that the requirements for a monolingual tokenizer are to be able to merge rule databases and for a multilingual tokenizer, to do this job dynamically.

4.1 The Standard Approach

Problems of tokenization include word tokenization and sentence tokenization. (Grefenstette and Tapanainen, 1994) lists all the problems arising in text tokenization (dates, references, numbers, accronyms, abbreviations, compound words, punctuation). They expose the standard approach to text tokenization based on lexical analyser generators using pattern-matching via regular expressions (e.g `lex` & `flex`).

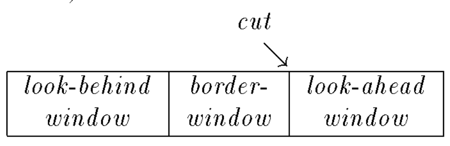
This solution is one of the most used and certainly one of the cleanest but it has some limits. The rules are declarative but *not interpreted*. They have to be compiled to generate an automaton in a source file. This source file has to be compiled and linked to the application. Moreover, *the rules must be ordered*, so including a new rule in a base is not so easy.

The state of the art does not enable us to solve our problem. In fact, we can't merge databases since the rules are ordered and even if we find a way to merge them, this cannot be done dynamically since the compilation makes it static. Moreover, the lack of flexibility prevent us from adding a new language without recompiling the whole application.

4.2 Improving the Standard Approach

To solve these problems, we consider the *dual* approach. In fact, we note that on a text it is often easier to locate tokens via their *border* with their environment rather than locating them by their constituents. So, instead of writting rules defining explicitly tokens as done in the standard approach, we write rules specifying borders: the tokens are implicitely defined as the region between two cuts.

A rule is a unique regular expression formed by three sub regular expressions: an optional left-context, a border-window, an optional right-context. The left-context and the right-context can be seen respectively as look-behind and look-ahead windows. The border between two tokens is defined between the border-window and the look-ahead window (i.e after the last character matched by the border-window).



The analysis is done in one pass from left to right. At the beginning of the process, we assume that a cut is located before the first character of the input.

A rule is applicable if at least one matching of the whole regular expression can be found in the input and if the first character matched by its border-window is after the last defined cut.

Among the applicable rules, the triggered rule is the one for which the first character matched by its border-window is the nearest character after the last defined cut. A new cut is defined after the last character matched by its border-window.

No more cut can be set before the last defined cut: all the applicable rules for which the first character of the matched border-window is before the last defined cut have their matching recomputed. This is done such that they become applicable or that no more match of the whole regular expression can be found.

At the end of the process and since the match of a look-behind window can start before a last defined cut, every matches of all regular expressions have been found.

Thus, the rules are independant and the rule databases are not ordered. This is possible thanks to both the dual approach and the rule triggering. The rules are totally declarative and interpreted, no order is required in the rule base.

4.3 Some remarks about the method

It is interesting to note that specifying tokens via their border is not really new.

The same approach is also used for the Penn TreeBank's tokenization with `sed` scripts. The standard unix command `sed` (stream editor) apply in sequence editing commands defined in a script on the input stream. The commands modify the input. Tokenization is done in inserting a special character (a border) in the stream.

The improvment of our method is the rule triggering combined with the semantic of the rules. This allows tokenization in one pass and the absence of order among the rules.

5 Conclusion

In this article, the problem we tried to solve was the tokenization of a multilingual text in the framework of Natural Language Diagnosis. In our earlier related work, an answer to Natural Language Diagnosis has been given but the tokenization of multilingual text was a problem since it was done by a french ad-hoc tokenizer.

Our goal was to find an elegant way of tokenizing multilingual texts. So, efficiency was not the first goal. The study of the problem in multilingual framework clarify the monolingual tokenization process and lead us to find a new way to solve it. We propose to separate generic rules and language specific rules. The consequences are clear. Each database become smaller and, thus, more readable. Adding a new language is quiet simple since we just have to write the few specific rules and to merge statically or dynamically the shared database. Updating a generic rule is also simple since one file has to be modified for all the languages.

Looking for a solution to multilingual text tokenization, we found an elegant architecture to tokenize monolingual text. Then, studying again the multilingual aspects of the problem, we find that it was possible to merge all the databases to tokenize multilingual texts. The condition was the absence of order among the rules in the databases.

Using the fact, that it is often easier to define tokens implicitely via their border rather than by then constituents, we proposed an algorithm to tokenize in one pass an input stream.

References

Emmanuel Giguet. 1995a. Categorization according to language: A step toward combining linguistic knowledge and statistic learning. In *4th International Workshop of Parsing Technologies*, Prague - Karlovy Vary, Czech Republic, September 20-24.

Emmanuel Giguet. 1995b. Multilingual sentence categorization according to language. In *Proceedings of the European Chapter of the Association for Computational Linguistics SIGDAT Workshop "From text to tags : Issues in Multilingual Language Analysis"*, pages 73–76, Dublin, Ireland, march.

Gregory Grefenstette and Pasi Tapanainen. 1994. What is a word, what is a sentence? problems of tokenization. In *3rd International Conference on Computational Lexicography (COMPLEX'94)*, pages 79–87, Budapest. Research Institute for Linguistics Hungarian Academy of Sciences.