ROBUST EXTENDED TOKENIZATION FRAMEWORK FOR ROMANIAN BY SEMANTIC PARALLEL TEXTS PROCESSING

Eng. Marius Zubac[1] and Prof. PhD Eng. Vasile Dădârlat[2]

[1]Department of Computer Science, Technical University, Cluj-Napoca, Romania
[2]Department of Computer Science, Technical University, Cluj-Napoca, Romania

ABSTRACT

Tokenization is considered a solved problem when reduced to just word borders identification, punctuation and white spaces handling. Obtaining a high quality outcome from this process is essential for subsequent NLP piped processes (POS-tagging, WSD). In this paper we claim that to obtain this quality we need to use in the tokenization disambiguation process all linguistic, morphosyntactic, and semantic-level word-related information as necessary. We also claim that semantic disambiguation performs much better in a bilingual context than in a monolingual one. Then we prove that for the disambiguation purposes the bilingual text provided by high profile on-line machine translation services performs almost to the same level with human-originated parallel texts (Gold standard). Finally we claim that the tokenization algorithm incorporated in TORO can be used as a criterion for on-line machine translation services comparative quality assessment and we provide a setup for this purpose.

KEYWORDS

Semantic Tokenization, Bi-lingual Corpora, Romanian WordNet, Bing, Google Translate, Comparative MT Quality Assessment

1. INTRODUCTION

When faced with the task of text processing today's systems need to implement robust procedures for input text acceptance no matter the final processing purpose ranging from information retrieval and sentiment analysis to machine translation applications. This acceptance criteria may include various filtering pre-processing of the text, however the vast majority of researchers [1], [2], [3] agree that tokenization should be the first step of the text processing pipe as shown in the Figure 1 preceding any lexical analysis step.

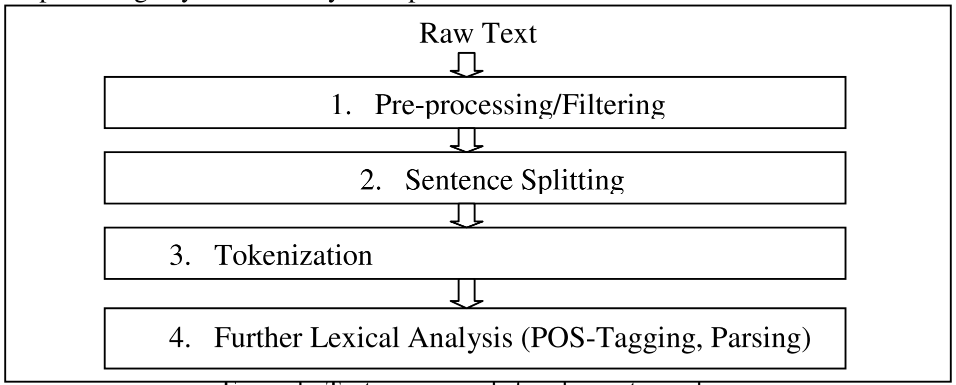
Raw Text
1. Pre-processing/Filtering
2. Sentence Splitting
3. Tokenization
4. Further Lexical Analysis (POS-Tagging, Parsing)

Figure 1. Text processing before linguistic analysis

The raw text that may very well originate from the internet in html/xml format is first extracted as plain text inside a process usually called *filtering*. Then the text is forwarded to the next process in the pipeline: the *sentence splitting* process. It is this process the one responsible with the sentence boundary detection. For issues related to this process good surveys are provided in [4] and [5]. In this paper however the focus is set on the word-level tokenization process.

2. TOKENS AND TOKENIZATION REVISITED

2.1. What is a token? What is tokenization?

While everyone agrees that tokenization is necessary there is no perfect agreement on what a token is on one hand and what the tokenization process is or should be on the other hand and all these for good reasons. The paragraph title paraphrases a well-known seminal paper by Grefenstette and Tapanainen "What is a word? What is a sentence?" already mentioned in Section 1 [3]. There the authors states that "**the isolation of word-like units** is called tokenization" and again in this process two types of tokens emerge: "one type corresponding to units whose character structure is recognizable, such as punctuation, numbers, dates, etc.; the other type being units which will undergo a morphological analysis". Many researchers start by analyzing the notion of word itself and in [1] Webster begins by looking at what is a word from a lexicographer's perspective. While "words", "collocations" and "multi-morpheme items" are significant from a lexicographer's perspective, the author recognizes that the concept of token is important from a NLP or MT application perspective and observes that a token cannot be broken into smaller pieces for the purpose of these applications. The token is then a **terminal node** [1]. Palmer in [5] expresses the fact that "tokenization is the **process of breaking up the sequence of characters in a text by locating word boundaries**, the point where one word ends and another begins". According to [6] there is "segmentation of a stream of signals, classification of these signals and relating those to linguistic units" vs. tokenization that refers to **segmentation of a written text**. In [7] the token is viewed from a linguistic status, a formal one and from an application-based perspective. While in [3] a certain morphological analysis is presumed to follow the tokenization process in [8] for instance, in *lexical analysis* processing for *information retrieval* applications "*tokens* are groups of characters with collective significance". More than that, the ones that do not carry significance and belong to *stoplists* and *negative-dictionaries* are even discarded as non-tokens. Applications centered on word sense disambiguation (WSD), text-mining, terminology extraction, (cross lingual) question-answering (QA), information retrieval, speech synthesis, etc. all have different needs than for instance machine translation (MT). To summarize an answer to the paragraph questions we conclude that each type of application defines in fact what a token is for its own purpose, and consequently adjusts the tokenization process definition as well. A practical consequence would be to build flexibility in the tokenization process in order to address different tokenization styles based upon the field of application.

2.2. Why is tokenization hard?

A simple answer to this question is: lots of issues to address, lots of ambiguities to solve. In addition more requirements shown in section 2.3.3 may complete the list.

2.2.1. Multiple issues to address

In [5] Palmer states that tokenization algorithms must address: **language dependence**, **character-set dependence**, **application-dependence** and **corpus dependence**. Specifically at the language dependence level the author observes that detecting word boundaries and solving the possible ambiguities in the process is language specific and further observes that written Chinese and Japanese although have adopted punctuation marks for end of sentences do not have word boundaries. Tokenization follows different types of algorithms studied in [10], [11] (maximum tokenization principle) and [12] (one tokenization per source). At the character-set dependence it is mentioned the character encoding problem and it is observed the large usage of the GB and Big-5 encoding for Chinese and notes that different rules must be in place for TIS620 versus UTF-8 character encoding for Thai. The automatic discovery of text language and encoding system is delicate, statistic-based and may occasionally point to a wrong language and/or character-set encoding. In our opinion a tokenizer should be given along with the text, both the language and the character-encoding set parameters. Historically, in case of Romanian language many (on-line) texts have been generated using ISO 8859-2 (Latin 2) character encoding using ş and ţ (with cedilla). It was the wide adoption of the Unicode standard and the UTF-8/16 encodings that paved the way for Romanian ș and ț (with comma) correct diacritics to come back to life again. Although correcting Romanian diacritics is trivial, the process of total recovery for most of the on-line texts must reach the semantic level of the processing as shown in [13]. Application dependence is explained in [5] by noticing that a token like *I'm* in English could be left as such or expanded to *I* and *am*, depending upon how these tokens would be analysed in the next processing step either a lexical or semantic analysis. Corpora dependence is related with the fact that tagged corpora used in many researches are tokenized in a certain way, for instance in [5] is mentioned that the word *governor's* is considered one token in the Brown corpus [14] while is two tokens in the Susanne corpus [15]. This is a tokenization issue only if using further other toolkits aligned to specific corpora is a requirement. Unfortunately there is no unique standard for tokenization, tagging, etc., and depending upon applications various large used corpora act as de facto standards. Ambiguities come in many flavours. There are punctuation ambiguities, special characters ambiguities and the ones generated by the words and expressions themselves. Another way to look at ambiguities is to recognize the "good-old/classical" ones versus the "new wave".

2.2.2. "Good-old/classical" ambiguities.

Punctuation related ambiguities. From a linguistic perspective one can define three alphabets [6]: a token alphabet, a punctuation alphabet and a delimiter alphabet. The main issue related with this approach is that these three alphabets are overlapping. The rule of thumb for tokenizing punctuation is to treat them as separate tokens. Palmer [5] observes that are many cases however when punctuation should be "attached" to another token. The author observes that punctuation is language specific and gives examples for English. In this paper a series of examples for Romanian will be also given to further illustrate the language specific treatment of punctuation. Almost always the **blank space** is a delimiter, but there are notable exceptions where the splitting by space is usually performed but not desired: in the multi-word expressions like *in spite of* that should count for a single token. Other noticeable situations where space should not be considered delimiter for tokenization, according to [6] are: German large numbers grouping digits delimiter, phone/fax prefixes, etc. All these situations can be detected and fixed by means of regular expressions-based rule processing. The **hyphen** (-) is almost always part of the (word) token alphabet [6] and is not to be confused with the dash character, although the more informal the text (like blogs, chats) the more chances for confusion. This character plays an essential role in *multi-part words* (see the appropriate paragraph) in which case splitting the multi-part word should not be desired but also in building *grammatical structures* like in French e.g. *va-t-il*, *celui-ci* or in Romanian, e.g. *ne-am dus*, *v-ați adus*, *maică-sa*, where splitting is necessary for tokenization. Other undesired splitting on hyphen occurs in number ranges, phone/fax numbers, calendar dates where regular expressions may help. Another source of confusion and ambiguity is generated by the fact that dashes are sometimes used instead of hyphens. The **comma** is usually a delimiter [6] except in German and American numerical expressions where it is part of the token. This situation can be handled with regular expressions. The **period** (.) character beside the important role as sentence delimiter (full stop) and as an abbreviation marker may be can be encountered in many other constructs like name initial, email and internet addresses, dates, etc. where regular expressions matching techniques can be employed. It can be ambiguous (along with comma) in numerical expressions, reason why detecting/passing the language indicator is crucial. The **apostrophe** is a word delimiter that beside the Saxon genitive that can be usually recognized and expanded, also serves for contractions like *I'm*, *he's*, *it's* and in forming some plurals like in *I.D.'s* or *1980's* [5]. In general contractions can be detected using table-lookup or regular expressions-based strategies but sometimes reconstructing the full constituents may be ambiguous. For example *he's* may be expanded to *he is* or *he has* or *l'* can be either *le* or *la* in French. In our vision reconstruction is necessary (maybe to just disambiguate other tokens) even if deeper analysis is necessary.

Word related ambiguities. This category of ambiguities concerns abbreviations, acronyms, named entities, multi-part words and multi-word expressions. In most of the cases (as well in Romanian) **abbreviations** are sequence of characters ended with a period. Recognizing and eventually expanding the abbreviations is essential for tokenization and look-up tables and lists based strategies may obviously help, however these techniques are not sufficient for some reasons like: abbreviations are "productive" [5]. There will often be "new", out-of-the-list abbreviations not detected by the tokenizer; they may come without period character like in *Mass* for *Massachusetts* (cited in [5]); the text will occasionally contain abbreviations from other languages. Other type of abbreviation ambiguities may arise in the abbreviation expansion process (e.g. should be *St.* expanded to *Saint*, *Street* or *State* - example from [5]). The most important abbreviation detection ambiguous situation may arise in cases where the abbreviated word is also the last word in the sentence. This type of situation is frequent and in [3] is stated that in the Brown corpus there are 48885 sentences and 3490 contains at least one non-terminal period (full-stop). In [3] the authors experiment with various filtering and both lexicon and non-lexicon based strategies to detect true abbreviations in a context where the text is not sentence-level segmented and reports a success of over 99.7% on the Brown corpus. In many of the situation shown above cannot be solved at this level and a deeper semantic analysis is required. **Acronyms** should be tokenized as such, without expansion, for both text and MT processing purposes. The discussion starts though when other tokens need to be disambiguated and the actual acronym meaning is required. In Table 1 are listed a few ambiguous acronyms and sometime when there are more meanings in the same semantic domain choosing the right one from the list is not trivial.

Table 1. Ambiguous acronyms.

Acronym	**Description**
AR	Aspect Ratio or Assault Rifle
CD	Coefficient of Drag or Compact Disc
HP	Horsepower or Hewlett Packard
PFD	Personal Flotation Device (boating) or Partial Fraction Decomposition (math)

To complicate even more some acronyms can be also valid abbreviations written without the ending period or just full regular words. If expansion is necessary acronyms can be first detected with regular expressions but then should be deferred to list-lookup strategies. Abbreviations and acronyms disambiguation in medical discourse are studied in [16]. **Named entities** disambiguation faces the same type of ambiguity and treatment as acronyms. A much mediatized name *Bush* may refer according to Wikipedia to one of the two presidents of United States, other Bush persons, the family name, eight places form US, Canada, UK and NZ, four brand names, and the name can be found in other NER constructs in cinematography, music and sports domains. Is worthwhile mentioning that even in the same text president Bush can be mentioned in various forms like *Bush*, *G. Bush*, *G.W. Bush*, *George Bush* and to automatically detect that the words refer to the same person is not trivial. Eventually the tokenizer should perform here a text-normalization. The named-entity recognition subtask is also vital for information extraction. The main objective of this subtask is the identification of proper names and also their classification into semantic categories (person, organization, location, artefact, etc.). It is estimated that around 20% of named entities occur in ambiguous positions. In relation with ambiguous situations concerning capitalized words and NERs in [17] is cited the example *Daily, Mason and Partners lost their court case*, where it is clear that *Daily, Mason and Partners* is the name of a company. However in *Unfortunately, Mason and Partners lost their court case*, the name of the company does not include the word *Unfortunately*, but the word *Daily* is just as common a word as *Unfortunately* [17]. In [5] the author observes that the process of deriving a new semantic or grammatical meaning from a group of words called **multi-part words** is specific to many written languages. This process may imply completely agglutinating the constituents like in Turkish, Swahili, or most Altaic languages and common to German (a non-agglutinative language) as well (e.g. *Nichtraucher*, non-smoker) or just grouping the constituents and separating them with hyphens in many European languages including English (e.g end-of-line, New York-based), French, or Romanian (e.g. *bine-crescut* (Eng. well behaved), *floarea-soarelui* (Eng. sunflower). If the tokenization algorithm separates the multi-part words by hyphens (as it normally should) these tokens must be reconstructed back using a table-lookup strategy. Beside multi-part words another multi word compound formation separated by spaces is the **multi-word expression** that although is made of multiple words has a precise meaning sometimes replaceable by a synonym and should be treated like one single token, In [5] are given English examples like *in spite of* which is equivalent with *despite* and expressions borrowed from other languages like *au pair*, *de facto*, *joie de vivre* which should again treated as a single token. In this category also belong dates that have the month expressed in a word format, dates having the day of the week, some numerical expressions (e.g. $2 million), etc. In this case the reconstruction of the token may imply a combination of regular expression-based rules and table-lookup strategies and may prove to be rather complex. Detecting a **multi-word token** raises a certain number of technical problems. While **continuous** multi-word tokens can be detected with a table-look-up matching algorithm in case of **discontinuous** multi-word token e.g. *keep* please *in mind* or *keep* for instance *in mind* that have a generic form of *keep* [V/NP*] in mind* algorithms must incorporate syntactic analysis, partial parsing and even more sophisticated knowledge processing according to [1].

2.2.3. The "new wave" ambiguities.

Lately we have all witnessed the large expansion of social networks. Blogs account for a large quantity of written text that is of great interest to NLP applications like social intelligence, data-mining, information retrieval and extraction, sentiment analysis to name just a few. A typical English blog is shown in Figure 2.

> Great job @justinbieber! Were SOO PROUD of what youve accomplished! U taught us 2 #neversaynever & you yourself should never give up either♥

Figure 2. Non-standard English text extracted from a blog

This text highlights some tokenization issues: special characters treatment: @, #, & or ♥; abbreviation detection and expansion issues related to tokens like U, 2; capitalization issue related to *SOO PROUD*; absence of apostrophe in contracted *you've*; grouping words to emphasize them *neversaynever*.

The Romanian blog text shown below in Figure 3 contains in addition: lack of capitalization at the beginning of the sentence; total absence of Romanian diacritics; frequent presence in text of foreign words (mostly English) like *link*, or *back*.

cred ca GORAN va fi rescris, sincer ma gandesc si la factorul marketing, doresc sa-l trimit catorva situri mari pentru un link back. poate o fi cineva care imi va aprecia balariile.

Figure 3. Authentic post from http://www.blog.betforcash.ro/2007/08/analiza-partea1.html

All these issues present more challenges for the tokenization process. In the new types of text presented above there are new sources of ambiguities, the most important being generated by lack of diacritics in Romanian texts.

2.3. Tokenization - a sense of direction

Tokenization is not a done deal yet, especially when concerned with performance, robustness, complex annotations, solving ambiguities in languages other than English. In [7] there are mentioned two factors that explain the renewed interest in tokenization: first the scale of NLP application has changed. "Toy grammars and lexica belong to the past". Very large corpora are constantly assembled and the web is processed constantly for indexing purposes. The other factor mentioned is the recently interest in glass-box (as opposed to black-box approaches) evaluation of text 'pre-processing' tasks, tokenization included. Using the *elephant in the living room* metaphor [18] the author states that tokenization "is a problem that is impossible to overlook whenever new raw datasets need to be processed or when tokenization conventions are reconsidered" and stresses that errors occurring early in the NLP pipeline affect further analysis negatively. While in case of Standard English tokenizers usually provide good quality output it is still much to be done in the case of social media text analysis. If for instance we try to pass the English blog text for translation to Romanian using the Google Translate on-line translation services (http://translate.google.com/) the result is shown in Figure 4.

Mare de locuri de muncă @ justinbieber! Au fost SOO mândru de ceea ce le-ați realizat! U ne-a învățat 2 # neversaynever si te-ai nu ar trebui să renunțe, fie ♥

Figure 4. Translation in Romanian of the text presented in Figure 2 using the Google Translate on-line services

Here the tokenizer performed the splitting by the book but failed to reconstruct and hence interpret correctly the tokens for translation purposes (@ was left split from the twitter username, emphasized SOO was not replaced with *so*, U was not understood as *you* and 2 as *to*, *neversaynever* was left un-tokenized and unrecognized). Another observed trend is the move toward more complex and meaningful annotation schemes, recognizing some limitations in the PTB tokenization style. In this paper we propose a more complex annotation mechanism based on the concept of token class.

2.3.1. The concept of token class

While researchers recognized the variety of tokens encountered in the text we express the opinion that this fact has not been studied and exploited enough for the benefit of the tokenization process. We consider that the concept of token class is intuitive and comes in hand to solve many aspects of tokenization. *Period*, *word* and *multi-word expression* are all valid and intuitive tokenization classes. The task of tokenization can be further specified as being the process of splitting the sentence in meaningful tokens and assigning each token to a certain class. From an ambiguities solving perspective the concept of class holds benefits as well paving the way that makes the disambiguation process (no matter how complex) clear and manageable. If for instance the tokenizer recognizes that for instance the token *li* is a valid roman numeral (51) belonging to this class (*roman numeral*) and also a valid Romanian pronoun belonging to class *word* as well, the tokenizer should mark in annotations the token to both classes, and should recognize an ambiguity situation. The decision of handling the ambiguity 'on the spot' or deferring it to the text analysis pipeline is another matter. We will further analyse the token class concept for Romanian and propose an algorithm for handling class-based tokenization in section 3.3.

2.3.2. Inline vs. stand-off annotations

Annotations may be created **inline** (in the same file) as or can be **standoff** [19] generating a file with pointers to the tokens. In [20] is given an example of inline mark-up for tokenization, shown here in Figure 5 containing also class attributes information for words **w**, numbers **n** and punctuation **p**.

```
<W c=w>It</W>▪<W c=w>was</W>▪<W c=p>'</W><W c=n>3</W><W
c=p>'</W><W c=p>.</W>
```

Figure 5. Sample of tokenized sentence reproduced from Mikeev [20]

The space between words (▪) in the initial text is preserved in between <W> xml elements. The standoff annotation is more complex, is always generated automatically, leaves the text intact (which may be very well a read-only source, or for security/copyright issues) and generates structures that may contain annotations in multiple layers that contain pointers to the original data. Standoff annotation has been successfully used in many projects like NITE [21], GATE [22], WHITEBOARD [23], DELPHIN [24, 25], XCES [26, 27] xComForT [28] or BLIS [29] for bilingual encoding.

2.3.3. Tokenization design requirements

Besides the obvious requirements for a tokenization algorithm of recognizing unambiguously tokens and sentence boundaries, here is an additional list of requirements for tokenization without claiming that we have exhausted them: text traceability; flexible text normalization; generating the tokenized output in XML format; inline vs. standoff annotation option; flexibility and trainability; aligned to popular corpora formats; logging; language detection; character-encoding handling/conversion; missing diacritics detection and restoration. It is often desirable (and most tokenizers implement this feature) for some text structures to be 'normalized' and as such to be changed. For instance the tokenizer from the Charniak & Johnson [30] parser makes modifications to the text lemmatizing expressions such as *won't* as *will* and *n't*. In [31] the author expresses the opinion that in this case mostly for inline annotation tokenizers it is critical to provide more than just an additional but **full traceability** from the token objects back to the raw text annotation and introduces the concept of *characterization* to express the character position links back to the source. It is important to have the tokenizer capable to work in three **normalization contexts**: without normalization, with (forced) normalization, and with tokenizer-driven text normalization, for each type of text normalization option (including abbreviation and acronym expansion, apostrophe marked genitive and marked contracted forms, etc.). Let's notice here that abbreviations and acronyms expansion may cause ambiguities. **Generating the tokenized output in XML format** requirement has not acquired the deserved attention in the research community and unfortunately there is no standard for the tokenization process output by itself and in this situation the role of de facto standard has been taken by the Penn Treebank (or **PTB**) [32]. While the **PTB** has a rich set of tags for POS tagging and further syntax analysis at the tokenizer level itself the format is rather Spartan. The GENIA corpus and project [33] states that following the PTB tokenization scheme GENIA uses one XML tag for both words and punctuation marks. It should be clear that in order to support abbreviation expansion, traceability, information about tokenization rule applied or ambiguity situations to be transferred to the syntactical analysis level a more elaborated xml structure must be elaborated. **Flexibility** is a requirement that concerns mostly rule-based tokenizer. It is important when faced with new texts to be able to quickly add new rules or change their execution. For statistical-based tokenizers this requirement is expressed by the **trainability** feature. **Aligning the output format to popular corpora formats** (but mostly with **PTB**) enables further processing to be performed with the corpora- related NLP toolkits. Although adding **logging** to the tokenization process may look like an unnecessary overhead, sometimes we might wonder about the tokenizer's decisions in classifying the tokens or in choosing the disambiguation strategy. This detailed information should be logged in standoff annotation and should be used in for text tokenization analysis and in tokenization debugging and development. **Language detection** is an important issue in tokenization for two reasons. We need to understand the language context of the sentence to be able to extract the proper language-specific rules from the database and to point to the proper dictionaries/lists of abbreviations, acronyms, multi-part-words, etc. A second reason for language detection within a sentence context lies in the fact that both formal and informal may often contain expressions and words from other languages (like French or Latin) than the text language. Romanian blogs are filled with English words like shown in Figure 3. Issues related with *Natural Language Diagnosis*, and language identification are studied in [34] and [35]. Although **character encoding detection and handling** is an important issue for some languages and definitely is a valuable feature added to the tokenization process we need to say that this operation can also be seen as a pre-processing or pre-tokenization step and can be treated separately. For certain languages like French, German, Arabic or Romanian that all have diacritic characters in their alphabets the ability for tokenizers to **recognize and restore the correct word with diacritics** from the form without diacritics is important and not trivial. At least for Romanian the diacritic characters have semantic implications and their absence increases artificially the number of *homographs* on which the disambiguation analysis must be performed. Lately the social web is the major source of text without diacritics and this situation will not change any time soon.

2.3.4. Tokenization algorithms, principles and trends

From a practical side according to [1] there are two aspects of tokenizing: 1) automatic segmentation (using dictionaries for example) and 2) strategies for disambiguation. Automatic segmentation algorithms can be divided in two main classes: rule-based and statistical based usually on **HMM** models. Both these classes may use lexicon resources to correct and validate tokenization. Overview of tokenization algorithms is beyond the scope of this paper and as well is algorithms for un-segmented languages like Chinese. In [10] the author introduces **critical tokenization** a precise formal way to define and discover various types of ambiguities, following the principle of maximum tokenization, defines critical and hidden ambiguities (*blueprint* vs. *blue print*) and proves that the principle of ***maximum tokenization*** [11] would not be effective in resolving the critical ambiguity in tokenization. In [12] Guo reports on the principle of ***one tokenization per source*** that states that if a text contains more sentences having the same ambiguity this must be resolved in the same manner along the whole text. In [9] it is proposed a rule-based **extended tokenization** process that interprets and groups isolated tokens to create *higher level tokens* including all sorts of linguistic knowledge (e.g., grammar rules, dictionaries). We agree we this approach and believe that it is the tokenizer's task to provide meaningful tokens and if needed even semantic analysis no matter how complex must be employed at this stage. In [17] the author claims a *Document-Centered Approach* that states that capitalized words and abbreviations occur at least once in an unambiguous context inside a document and that these values can be used further for the same tokens when these tokens appear in ambiguous contexts.

2.3.5. Strategies for disambiguation

Strategies for tokenization disambiguation can be grouped in two main categories: deferred and solved in situ. A first approach for handling complex ambiguities is to defer them to the downstream processes that will eventually solve them through lexical or syntactical analysis. An example of inline treatment of ambiguities is cited from [7] where is shown that in the Intex system [36] these ambiguities are treated like linearized graphs in Figure 6.

```
Luc a travaillé pour la [1 Ministère [2 de l'intérieur 2] 1]
```

Figure 6. Intex system inline tokenized senetence for "Luc a travaillé pour la Ministère de l'intérieur" [7]

In [24] the author defines a lattice for token ambiguities representation in standoff annotation and promotes the idea of passing the ambiguities for downstream processing. Regarding ambiguities we hold the opinion that these must be solved in situ. At this point we are not thrilled to use statistics and give a final verdict but rather invoke a complex mechanism for **semantic tokenization** described later in this paper in section 4.

2.3.5. Tokenizers evaluation

There are two important aspects to this topic. One regards the case when a certain tokenization process is designed to serve later in a pipeline for a specific NLP application. In this case it is important to measure the progress in the tokenization development process. Usually in this case the tokenization process is regarded as a classification algorithm where various tokens must be assigned to tokenization classes as discussed above in 2.3.1. In this case we can use the precision, recall, F1-measure and coverage set of indicators. **Precision** (or a**ccuracy**) is defined as the percentage of tokens returned by the algorithm that occurred in the hand segmented text (the Gold standard) in the same position and having the same class as the corresponding hand segmented token (1).

$$(1)\quad \mathbf{P} = \frac{\text{Nr of tokens from tokenizer found in Gold with the same class}}{\text{Nr of tokens offered by tokenizer}}$$

Let us mention here that close related with precision is also the **error rate** defined by the percentage of incorrect assigned tokens from the total number of tokens offered by the tokenizer in (2).

$$(2)\quad \textbf{Error rate} = 1 - P$$

Recall is the percentage of tokens and classes in the manually segmented text (the Gold standard) identified by the segmentation algorithm (3).

$$(3)\quad \mathbf{R} = \frac{\text{Nr of tokens from tokenizer found in Gold with the same class}}{\text{Nr of tokens in the Gold standard (expected)}}$$

The **F$_1$-measure** is the harmonic average of the measures (1) and (3) above, defined by the formula (4).

$$(4)\quad \mathbf{F_1\text{-}measure} = \frac{2PR}{P+R}$$

Also useful can be the **coverage** indicator (5) defined as the percentage of tokens generated by the tokenizer from the total number of tokens from the hand segmented text (the Gold standard).

$$(5)\quad \mathbf{C} = \frac{\text{Nr of tokens from tokenizer}}{\text{Nr of tokens in the Gold standard (expected)}}$$

From a methodological perspective a hand-segmented file/corpus is created and is set as a Gold standard file/corpus. This file is compared then with the tokenized file generated by the segmentation algorithm. An annotation guideline can be provided to the human annotator to bring consistency to the process.

A second aspect corresponds to the situation when multiple tokenizers are compared for the result of their output. In [7] it is shown the need for evaluation, and the difficulty of the endeavor. By analyzing 18 tokenizers within project GRACE [37] the significant discrepancies in the number of words and sentences detected is partially explained by the structural differences amongst tokenizers and also by the lack of standards starting with the concept of token. In [31] tokenization results from different tokenizers have been compared with a Gold tokenization but here the comparison has been facilitated by the common **PTB** output format standard that all tokenizers observed. The author introduced yet another measure, the total Levenshtein distance calculated over the sentences tokenized differently than in the Gold standard.

3. THE TORO TOKENIZER SYSTEM

3.1. Overview and architecture

The decision for building **TORO** (**TO**kenizer for **RO**manian) was made upon several considerations. There are several generic tokenizers that have been used for tokenization of Romanian like MtSeg [38] used in [39], GATE [22], or RO-Balie [40] and specialized tokenizers like the tokenization web service hosted by **RACAI** (Research Institute for Artificial Intelligence of the Romanian Academy) web site [41]. Although some of these tools obtained good results in practice we felt the need to develop a more robust specialized tokenizer for Romanian. A second requirement was to be able to use this tokenizer in a complete Romanian-oriented NL text processing pipe-line in a context which is both stable and controllable. From a software design perspective the TORO System has been built in a modular structure as part of a more generic NLP framework - **PERSEUS** that contains also modules for lemmatization, POS-tagging, Word-Sense Disambiguation, and Diacritics Restoration. The main components of the tokenization system are shown in Figure 7: the TORO tokenization module; Tokenization Class Disambiguation Module; tokenization resources management system; the MRDEX lexicon; evaluation utilities; conversions utilities; external services access interface.

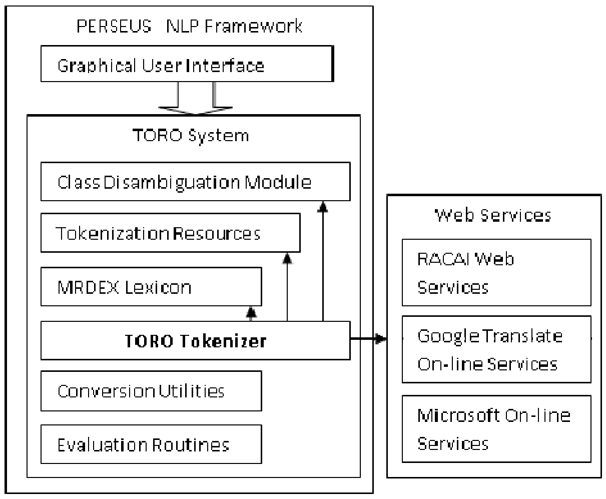

Figure 7. PERSEUS framework – TORO System components

3.2. The tokenization module. Word-entities

The tokenization module is responsible with the actual tokenization process. TORO uses a rules-based approach, generates inline annotation in the xml format, permits reconstruction of the initial text, uses lexicon resources (MRDEX) and lists for abbreviations, acronyms, named entities, and clitics. The tokenizer employs the concept of tokenization class and when ambiguous situations occur, it invokes the Tokenization Class Disambiguation Module for the task (section 4). The TORO algorithm works based upon a set of presumptions regarding the input text. The input text is a bitext (Rou-Eng), in the sense that there are two sentence-level aligned corresponding text files: one for Romanian and one for English. The files are sentence-level segmented. Each line contains one sentence/segment (both for English and Romanian).

The Romanian text contains diacritic characters. For disambiguation purposes there is access to: English tokenization services; on-line access to RACAI/Ro-WordNet; on-line translation services. While at a first look these requirements seem quite restrictive and limiting the usefulness of the tokenizer, in reality these presumptions are not hard to meet as we will prove in the paper. The main tokenization algorithm is a complex and acts in four stages. Each segment S_j for the source language (Romanian) is analyzed and split into entities. The stage I of the algorithm is presented

Each segment S_j is considered a string of characters containing spaces (or tab characters).
1. S_j is segmented by space and a first list of entities $L0_j$ is obtained. Each entity e_i from $L1_j$ is considered to have the format:
e_i = lc_i+lp_i+we_i+rp_i+rc_i where lc_i – left special chars; lp_i – left punctuation; rp_i – right punctuation; rc_i – right special characters; we_i – the word entity; lc_i,lp_i,rp_i,rc_i can be eventually the null strings; the ‚+' is the string concatenation operation.
2. In step 2 we build a list $L1_i$ that has all the lc_i,lp_i,rp_i,rc_i and we_i tokens.

Figure 8. TORO tokenization algorithm – stage I

3.3. Tokenization classes

At this point we introduce the concept of **tokenization class**. We are not satisfied with the reduced set of {word, number, punctuation} set encountered so far and propose a reach set shown in Table 2.

Table 2. Main TORO tokenization classes.

Class symbol	Class name	Class detection strategy	Additional notes
AC	Acronym	Table lookup+expansion	Includes POS-tagging info.
D	Dates	Regular expression	Standardization
E	Email address	Regular expression	Includes POS-tagging info.
NE	Named entity	Table lookup	Includes POS-tagging info.
P	Punctuation	Regular expression	Retains the character code
RN	Roman numeral	Regular expression + Table lookup	Some tokens like li, ci, mi, vi, vii are also tokenized at class W
U	Url address	Regular expression	Includes POS-tagging info.
W	word	Regular expression	

The strategies for classifying the word entities according to these classes use regular expressions matching, lists and table-lookups and combination of these two.

In the second stage of the algorithm presented in Figure 9 for each word entity we_i we apply in order all the regular expressions-based class detection rules from the rules database and obtain a possible set of overlapping class tags C_{wei}.

```
for each $we_i$ from $L1_j$
1. we apply all detection rules for dates, email addresses, IP addresses, numbers, etc and obtain a set of classes $C_{wei}$
2. if $C_{wei} = Ø$ then $C_{wei} = \{W\}$
3. if $Card(C_{wei})=1$ and $C_{wei} \neq \{W\}$ the token is final.
4. word entities are copied in $L2_j$.
end for
```

Figure 9. TORO tokenization algorithm – stage II – regex class allocation

Some examples of regular expressions for token class detection are presented in Table 3.

Table 3. Regular expressions for class detection

Class symbol	Regular expression	Additional notes
D	^(0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])[- /.](19|20)\d\d$	Dates in mm/dd/yy format
E	\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b	Email address
N	([0-9]+[,])*[0-9]([.][0-9]+)?	English number

Number detection is subject to regular expression matching with the observation that English, French, or Romanian for that matter, all require different regex formulae and therefore detecting or sending as parameter the language of the text is crucial. Emoticon and other Twitter-related tokens are examined on the site http://sentiment.christopherpotts.net/tokenizing.html. It is considered that emoticons are not only extremely common in social media but they are reliable carriers of sentiment and therefore important. Usually at this stage there are no class ambiguities with one noticeable exception. Some roman numerals like li, ci, mi, vi, vii are also valid Romanian words. In these particular cases $C_{wei} = \{RN, W\}$. A similar, ambiguous situation for English should arise from tokens like *2* and *4* which are usually numbers but in informal texts these tokens can stand for the words *to* and *for*. Obviously this is an ambiguous tokenization that can only be solved by invoking the semantic analysis algorithm mentioned in the fifth stage of the algorithm and described in details in section 4.

3.4. Word class tokens refinements. Periods, hyphens, apostrophes

In the third stage (Figure 10) the tokenizer analyses all the word entities assigned to the generic class W that contains periods (.), hyphens (-) and apostrophes (‘). Periods are sure signs of abbreviations. Hyphens and apostrophes are markers for clitics, and multi-word tokens. The separator role of hyphens inside the tokens is determined using lists and table-lookup strategies.

```
for each $we_i$ from $L2_j$ in class W and not final:
1. if $we_i$ ends with period then processAbbrev($we_i$) and
2. if $we_i$ contains hyphens or apostroph processHyphen($we_i$)
3. expanded abbreviations and hyphenated word entities are copied to $L3_j$.
end for
```

Figure 10. TORO tokenization algorithm – stage III – periods and hyphens

This process is also prone to generate ambiguities if multiple ways to expand the abbreviations are possible. In addition another ambiguity is presented by the last word in a period ending sentence: is it just a normal word or is it also an abbreviation? Examples of abbreviations are shown in Table 4, and we notice here that the abbreviation *abr.* stands for both *abreviere* noun and *abreviat* adjective generating expansion ambiguity. Examples of Romanian clitics are listed in Table 5 and multi-word tokens are presented in Table 6.

Table 4. Romanian abbreviation expansion list

Abbreviation	Expansion	English translation
Adj.	Adjectiv	Adjective
Anat.	Anatomie	Anatomy
Cont.	Contabilitate	Accountancy
Min.	Mineralogie, minerit	Mineralogy, mining

Beside the list lookup strategy for abbreviation expansion when a certain presumable abbreviation is not found in the list the algorithm employs a lookup in the MRDEX lexicon and returns a ranked list with possible candidate words for the given abbreviation.

Table 5. Common Romanian clitics

Clitic particle	Description
într-	Preposition in prefix position in *într-un* (in)
le-	Pronoun in prefix position in *le-ar* or *le-a* (vb. construct)
ne-	Pronoun in prefix position in *ne-ar* or *ne-a* (vb. construct)
s-	Pronoun in prefix position in *s-ar* or *s-a* (vb. construct)
-o	Pronoun in suffix position in *într-o*, or *dintr-o* (in)
-mi	Pronoun in prefix position in *să-mi* (vb. construct)

According to the Romanian orthography the hyphen (Romanian *cratima*) plays a complex role, one of them being a separator in compound words not completely welded.

Table 6. Romanian multi-word tokens

Compound words	English translation
câine-lup	Wolfhound, wolf dog
dus-întors	round trip
floarea-soarelui	sunflower
româno-american	Romanian-American
nou-născut	new-born

3.5. Word class tokens refinements. Capital letters, acronyms, named entities (NE)

In the fourth stage (Figure 11) the tokenizer analyses all the word entities that begin or are formed entirely with capital letters. This category contains important classes like acronyms, named entities (or **NE**s, e.g. name of persons, places, artifacts, etc.) and also presents an interesting ambiguity problem: is the first word in the sentence (starting with a capital letter) an ordinary word or is it a **NE**?

```
for each we_i from L3_j in class W and not final:
1. if we_i contains all caps then processAcronyms(we_i). Identified acronyms are marked final.
2. if we_i contains only the first letter cap. processNE(we_i,i)
3. word entities are copied to L4_j.
end for
```

Figure 11. TORO tokenization algorithm – stage IV – acronyms and **NE**s

Acronyms fall in two categories. A first category contains English-origin internationally recognized acronyms used as such in every language (e.g. CPU, RAM, CD, DVD, ADA). A second category of acronyms are language and country specific. These types of acronyms should provide a translation. In both cases acronyms are just identified and pointers to the list should be copied in $L4_j$. Examples of acronyms are listed in Table 7.

Table 7. Acronyms encountered in Romanian texts

Acronym	**Description**
CPU	Central Processing Unit
CJ	Cluj County (Romania)
SRI, S.R.I.	Romanian Information Service
UTCN, U.T.C.N.	Technical University of Cluj-Napoca, Romania

Acronyms are highly ambiguous (like shown in section 2.2.3 Table 1) many of them designating tens of different concepts. General ontology domain information can be attached to acronyms but this is useful only when there are domain indicators for the tokenized text as well. Named Entity recognition (NER), a task specific to information retrieval domain is solved in TORO through lookup strategies. A sample list of NEs for Romanian space is shown in Table 8.

Table 8. Romanian named entities from *Romanian Constitution* corpus

NE	**Description**
1 Decembrie	December 1st , Romanian national day
Avocatul Poporului	Ombudsman
București	Bucharest, Romania's Capital
Camera Deputaților	Chamber of Deputies
Consiliul Superior al Magistraturii	The Superior Council of Magistracy
Consiliul Suprem de Apărare a Țării	Supreme Council of National Defence
Curtea Supremă de Justiție	Supreme Court

TORO uses a list of over ten thousands named entities including names of persons, geographical places, names of organizations, famous literary and artistic work, etc. all related with the Romanian cultural space. A phenomenon typical for NER is token formation based on the maximum expansion principle. Let's examine the construct from Figure 12:

```
Curtea Supremă de Justi□ie a României
Supreme Court of Romania (Eng.)
```

Figure 12. Romanian NE construct *Curtea Suprema de Justiție a României*

This named entity word formation can generate multiple sensible tokenizations like the ones shown in Figure 13.

```
<Curtea Supremă><de><Justi□ie><a><României>
<Curtea Supremă de Justi□ie><a><României>
<Curtea Supremă de Justi□ie a României>
```

Figure 13. Possible tokenizations for Romanian NE *Curtea Suprema de Justiție a României*

From all the possible tokenizations only the last one is really the one we want. The *maximum tokenization principle* calls for creating the token with the maximum number of words while the construction refers to one single named entity. The lookup strategy starts looking for the whole chain of proper names and dropping gradually words from right to left until a match is found.

3.6. Word class tokens processing. Looking for multi-word expressions

In the fifth stage (Figure 14) the tokenizer analyses all the word entities that are not final and tries to identify multi-word expressions with database-lookup strategies. A new list L5 is compiled containing multi-word expressions assigned to class *M*. Some rules for detecting these expressions include the equivalence with one-word synonym (*câte o dată* , synonym *uneori* engl. sometime, *de bunăvoie*, synonym *benevol* engl. voluntarily), numerals (*douăzeci și unu* = 21) or in compound words constructions where the sense is somehow different from the sense derived from constituents (*Anul Nou* engl. New Year, *Evul Mediu* engl. Middle Ages).

```
for each we_i from L4_j in class W and not final:
1. processMultiWorddExpression(we_i)
2. multi-word expressions found and left word entities are copied to L5_j.
end for
```

Figure 14. TORO tokenization algorithm – stage V – Looking for multi-word expressions

For each detection of a multi-word expression a new tokenization is created and such a new ambiguity is inserted in L5 list. The last stage (VI) of the tokenization process is reserved for the disambiguation process described in details in the next section.

4. The Tokenization Class Disambiguation Module

The ambiguities discovered so far by the tokenization process are forwarded for resolution to the Tokenization Class Disambiguation procedure that provides deep semantic analysis for this purpose. One of the most employed methods when semantic is involved is based on the Lesk algorithm which keeps the resolution paradigm in a monolingual context. In this paper we describe method based on parallel texts processing. It has been mentioned that a presumption for this algorithm is to have a parallel text. The idea of this algorithm presented in Figure 15 is to use lexicon resources (MRDEX for Romanian text), off-line or on-line translation services and eventually the *Ro-WordNet* service (hosted by **RACAI** institute) to create translated tokenizations of the ambiguous situations and then detect from this set the one that matches the tokenization of the target parallel segment. The MRDEX lexicon is a resource created and maintained in-house and is based initially upon the DEX dictionary offered in a GNU-GPL license on the **DEX-ONLINE** website (http://dexonline.ro).

```
if segment S_j contains ambiguous tokenizations then
    1.  extract corresponding segment T_j from the parallel text:
    2.  for each we_ik that is ambiguous (there are k alternatives)
        2.1 get the lemma and POS from MRDEX: wi_ek= getMRDEXInfo(we_ik)
        2.2 translate lemma using on-line translation services (Google)/RACAIWdNet:
        tw_ek=getTranslatedLemma(wi_ek)
        2.3 tokenize the target segment T_j using RACAItoken on-line service.
        2.4 scan through tw_ek tokens and identify the one that is also in T_j tokenization.
        end for
end if
```

Figure 15. TORO tokenization algorithm – stage VI – Class Disambiguation Module

The **MRDEX** (**M**achine **R**eadable **DEX**) lexicon contains a changed table structure; it contains POS tagging information for each *word-form* using a tagset aligned to the **MULTEXT** standard for Romanian. The **MUTEXT** project initiative is presented in [42], and the POS tagset specifications are described in [43] and [44].We used in this algorithm the linguistic services hosted on the web services portal of the RACAI site - www.racai.ro. The web services are also described in [41]. We wish to thank this institution for the access to these services. In essence these services are: tokenization services for both Romanian and English texts, Romanian WordNet services, and Ro-En translation services. We have also experimented with other off-line and on-line translation services including Google Translate and Microsoft Bing. Let us reconsider a few ambiguities mentioned so far in this paper and see how the algorithm handles them. We have mentioned at the end of the paragraph 3.3 that a word like li is both a roman numeral (51) tokenized at class RN and also pronoun tokenized at class W. In case *li* has the sense of roman numeral the parallel text would also contain the word *li* which is not translated. In case when the li word is a pronoun the parallel text should contain one of the translated tokens for *li*: *they*, *their*. An example of text containing the word li is extracted from the 1991 Romanian Constitution document published also in English and French on the Romanian Deputies' Chamber website hosted on (www.cdep.ro) and shown in Table 9.

Table 9. Text disambiguation example – Roman numeral or pronoun?

No	**Source**	**Text with ambiguous token *li***
1	Romanian Constitution - Article 110, paragraph 4, Romanian text	Dacă **li** se solicită prezența , participarea lor este obligatorie .
2	Romanian Constitution - Article 110, paragraph 4, English text	If **they** are requested to be present , participation shall be compulsory .
3	Romanian Constitution - Article 110, paragraph 4, Microsoft Bing translation	If you are requesting **their** presence, participation is mandatory.
4	Romanian Constitution - Article 110, paragraph 4, Google Translate	If requested, and **their** presence is mandatory.

The Romanian text is presented in the first row. The English parallel text shown in the second row contains the word they that solves the ambiguity to class *W*. It is here the moment to comment on the parallel text existence presumption. We know that this is hardly the case. What rows three and four from Table 9 above show is the fact that for tokenization purposes we can rely on the translation output given by these first-class on-line translation services: Bing and Google Translate. Although both translations are not accurate they are good enough to serve the purpose. Let's illustrate this assumption with one more example. Another ambiguous situation is presented in the Romanian text shown in Table 10 first row. Here the word *Continental* is capitalized because it starts the sentence. This word can be tokenized as both normal adjective word to class W but can also be interpreted as the airline company Continental and tokenized at class NE (named entity). By invoking the Google Translate services we got the word *Continental* in a context that matches semantically the named entity which again solves the ambiguity to class NE.

Table 10. Text disambiguation example – Capitalized word – Normal word or Named entity?

No	**Source**	**Text with ambiguous token *Continental***
1	Romanian text - informal	Continental și-a amânat zborurile.
2	Google Translate	Continental has delayed flights.

A comparison between the TORO tokenizer and the similar service offered by RACAI is difficult to produce due to differences in NE lists used by the tokenizers. However on the Romanian Constitution corpus containing 10219 tokens, tokenized by hand and used like Gold standard for TORO development we measured a precision of 98.87% for TORO and below 80% for RACAI web service.

5. Romanian-English Machine Translation Quality Evaluation Criteria Based on TORO

The TORO algorithm presented in this paper suggests itself as a mechanism to be used in a setup for comparing machine translation services (MT) – translation quality. From the very beginning we must specify that this comparison is valid only for the Romanian-English language pair. The algorithm is presented in Figure 16.

1. A text *T* containing as many ambiguities as possible is first compiled for Romanian language.
2. This text is hand-tokenized to become the Gold standard *G*.
3. Then TORO is used for tokenization connected to the machine translation *A* service to handle ambiguities. The output is MT_A.
4. Calculate precision, recall, F_1 metrics comparing MT_A with *G* and build a global index I_A(P,R,F).
5. Connect TORO to the machine translation *B* and repeat steps 3 and 4.
6. Compare Indexes I_A(P,R,F) and I_B(P,R,F).

Figure 16. TORO – based setup for machine-translation quality evaluation

There are two levels of assessment for on-line translation services. The first and direct level is when the translation forces the disambiguation mechanism to create a different tokenization class for the ambiguous word/token than the class found in the gold standard file. A good example of this situation is related with interpreting roman numerals vs. pronouns as shown already in Table 9. Situation that might generate different tokenization.

A second level where the translation services may generate differences for the tokenization process is when the services return the same word class but with different POS or word sense information that is further needed to disambiguate other tokens in the sentence. The following examples analyse several homonymy situations encountered in Romanian language. A first example takes into account the homonymy presented by the Romanian word *broască* that has multiple meanings like *frog/turtle* or *lock.* The Romanian text in Table 11 line 1 is rather ambiguous and both frog and lock can be valid interpretations if no further context is provided. Both on-line services Bing and Google Translate had no doubts and translated to frog. If however we want to point that the lock was broken in line 4, only Google feels the different sense of the word *broască* and changes the translation to the correct one.

Table 11. Text disambiguation example based on homonymy of word *broască/frog vs. lock*

No	Source	Text with ambiguous word *broască*
1	Romanian text	El văzu broasca şi plecă.
2	Bing translation	He saw turtle and went away.
3	Google Translate	He saw the frog and left.
4	Romanian text	El văzu broasca stricată şi plecă.
5	Bing translation	He saw the wanton and frog went away.
6	Google Translate	He saw the broken lock and left.

Another interesting homonymy situation that can generate ultimately differences in tokenization class and therefore in the precision index is shown in Table 12, where the ambiguous word *nouă* has both the sense of *new* (adjective-word class) and *nine* (numeral- number tokenization class).

Table 12. Text disambiguation example based on homonymy of word *nouă/new vs.nine*

No	Source	Text with ambiguous token *nouă*
1	Romanian text	Du-te la poarta nouă.
2	Bing translation	Go to the new gate.
3	Google Translate	Go to the new gate.
4	Romanian text	Du-te la poarta nouă, nu la zece.
5	Bing translation	Go to the new gate, not ten.
6	Google Translate	Go to gate nine, not ten.

The Romanian text is ambiguous but both translation services return the translation based on the *new* meaning. In Line 4 though the text emphasized the sense of *nine* vs. *ten* and this situation has only been captured by the Google Translation. In the end if we decide to normalize *nine* to a number class token, Google Translation precision index would be higher. Based on a limited number of samples we cannot draw a clear conclusion but we showed the way in which TORO can be used to generate indexes for the translation services. The real issue we see with this approach is the fact that in order to draw a meaningful conclusion we need to build a large-enough and equilibrated corpus to act as Gold standard.

6. Conclusion and Future Works

In this paper we have presented the TORO tokenizer build for tokenizing Romanian texts. This tokenizer is robust, has a very high precision and can provide quality tokenization for next text processing steps in the pipeline. This tokenizer uses token class concepts, uses semantics and parallel texts to solve the tokenization ambiguities. In the absence of these parallel resources we proved that by invoking first-class on-line translation services for the disambiguation task we get almost the same information like the one retrieved from the parallel text. In addition we provided a setup based on TORO that can serve for quality translation comparison between various machine translation systems or services. Future work is required to test the impact of missing diacritics in the source text, and also for including the sentence boundary detection in the algorithm. Another area for continuous improvements with direct impact in the tokenizer's robustness regards feeding the tokenizer with a larger variety of text inputs mostly from the social media resources and continuously expanding the Named Entities, abbreviations and the acronyms lists. The experiments presented in this paper may be carried out for English-Romanian language-pair direction as well and based upon the experimental results using the on-line MT services (instead of parallel text resources) additional MT translation quality indexes can be produced.

References

[1] Webster, J.J. & Kit, C., "Tokenization as the initial phase in NLP", University of Trier, volume 4, pp. 1106-1110, 1992.

[2] Jurafsky, D., Martin H. J. 2009, Speech and Language Processing, Second Edition, Pearson Education International, Upper Saddle River, New Jersey.

[3]Grefenstette, G. & P. Tapanainen (1994). “What is a word, what is a sentence?” In Proceedings of the 3rd International Conference on Computational Lexicography (COMPLEX-94). pp. 79-87.

[4] Read, J., R. Dridan, S. Oepen, and L. J. Solberg (2012). “Sentence Boundary Detection: A Long Solved Problem?” The 24th International Conference on Computational Linguistics (Coling 2012). India.

[5] Palmer, D. D. (2000). “Tokenisation and Sentence Segmentation”. In R. Dale, H. Moisl & H. Somers (eds.), Handbook of Natural Language Processing, New York: Marcel Dekker, pp. 11–35.

[6] Lemnitzer, L. Tokenization, Seminar für Sprachwissenschaft, Eberhard Karls Universität Tübingen.

[7] Habert, B., Adda, G., Adda-Decker, M., Boula de Marëuil, P., Ferrari,S., Ferret, O., Illouz, G., Paroubek, P. (1998) Towards Tokenization Evaluation Proceedings of the First International Conference on Language Resources and Evaluation (LREC) Granada, May 1998.

[8] Fox, C., “Lexical analysis and stoplists”, pp. 102-130, 1992.

[9] Hassler, G. Fliedl, “Text Preparation through Extended Tokenization”, In Data Mining VII: Data, Text and Web Mining and Their Business Applications.Volume 37.WIT Press/Computational Mechanics Publications;Zanasi, A and Brebbia, CA and Ebecken, NFF 2006:13-21.

[10] Guo, J., “Critical tokenization and its properties” In Computational Linguistics, 23(4), pp. 569-596, 1997.

[11] Guo, J., “Longest Tokenization” In Computational Linguistics and Chinese Language Processing, 2(2), pp. 25-46, 1997.

[12] Guo, J., “One tokenization per source”. In Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics, 2(2), pp. 457-463, 1998.

[13] Dădârlat V., Zubac M., A Semantic Approach to Automatic Diacritics Restoration, In ACAM Scientific Journal Volume 20/2, 2011.

[14] Francis, W.N. and Kucera, H. (1982). Frequency Analysis of English Usage. Houghton Mifflin, Boston.

[15] Sampson, G. R., English for the Computer. Oxford University Press, 1995.

[16] Pakhomov, S., Pedersen, T., Chute, CG. Abbreviation and acronym disambiguation in clinical discourse. AMIA Annu Symp Proc. 2005:589–93.

[17] Mikheev, A., Periods, Capitalized Words, etc. in Computational Linguistics, 1999, vol 28, pp. 289-318.

[18] Evang, K., Basile, V., Chrupała, G., and Bos, J. (2013): Elephant: Sequence Labeling for Word and Sentence Segmentation. Proceedings of the EMNLP 2013: Conference on Empirical Methods in Natural Language Processing, Seattle, United States (unpublished).

[19] H. Thompson and D. McKelvie. 1997. Hyperlink semantics for standoff markup of read-only documents. In Proceedings of SGML-EU-1997.

[20] Mikheev, A., Text segmentation. In The Oxford Handbook of Computational Linguistics, Ed. Mitkov R., Oxford University Press, 2009.

[21] J. Carletta, J. Kilgour, T. O’Donnell, S. Evert, and H. Voormann. 2003. The NITE Object Model Library for Handling Structured Linguistic Annotation on Multimodal Data Sets. In Proceedings of 3rd Workshop on NLP andXML, NLPXML-2003.

[22] H. Cunningham, D.Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In Proceedings of the 40th AnniversaryMeeting of the Association for Computational Linguistics (ACL’02), Philadelphia.

[23] A. Frank, M. Becker, B. Crysmann, B. Kiefer, and U. Sch¨afer. 2003. Integrated shallow and deep parsing: TopP meets HPSG. In Proceedings of ACL-2003, Sapporo, Japan.

[24] Waldron,B., Copestake, A., 2006 - A Standoff Annotation Interface between DELPHIN Components. The fifth workshop on NLP and XML: Multi-dimensional Markup in Natural Language Processing (NLPXML-2006, 2006).

[25] Waldron, B., Copestake, A., Schäfer, U., Kiefer, B. (2006). Preprocessing and tokenisation standards in DELPH-IN tools. Proceedings of the 5th International Conference on Language Resources and Evaluation LREC-2006 (pp. 2263 – 2268). Genoa, Italy.

[26] Ide N., Romary, L. Encoding Syntactic Annotation in Natural Language Engineering, Volume 10 Issue 3-4, September 2004, Pages 211 – 225, Cambridge University Press New York, NY, USA.

[27] Ide N., Bonhomme, P., Romary, L. (2000) XCES: An XML-based Encoding Standard for Linguistic Corpora, Proceedings of the Second International Language Resources and Evaluation Conference. Paris: European Language Resources Association , 2000.

[28] Freese, M., Heid, U., Emele M. Enhancing XCES to xComForT - An Extensible Modular Architecture for the Annotation and Manipulation of Text Resources. 2003 Proceedings of the 3rd Workshop on NLP and XML, ECAL 2003, Budapest Hungary, April 2003.

[29] Kit, C., Chan, H.T., Liu, X. Encoding Hierarchical Bilingual Texts of Hong Kong Laws with XCES. Proceedings of The First International Conference and Global Interoperability for Language Resources, ICGL 2008 Hong Kong.

[30] Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (pp. 173–180). Ann Arbor, USA.

[31] Dridan, R., Oepen, S. 2012, Tokenization: Returning to a Long Solved Problem. A Survey, Contrastive Experiment, Recommendations, and Toolkit. Jeju, In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) pages 378–382, Jeju Island, Korea. Association for Computational Linguistics.

[32] Marcus, M.P., Santorini, B., and Marcinkiewicz, M. A. (1993), Building a large an annotated corpus of English: The Penn treebank, Computational Linguistics, 19(2), 313-330.

[33] Kim, J.D., Ohta, T., Teteisi, Y., Tsujii, J. GENIA Corpus Manual Encoding schemes for the corpus and annotation, Date of Release: 15 November 2006 , © Copyright 2006 Tsujiilab, University of Tokyo.

[34] Giguet, E., 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”, Dublin, 1995.

[35] Giguet, E.The Stakes of multilinguality: Multilingual Text Tokenization in Natural Language Diagnosis. Proceedings of the 4th Pacific Rim International Conference on Artificial Intelligence Workshop “Future Issues for Multilingual Text Processing”, Cairns, Australia, 1996.

[36] Silberztein, M., (1993). Dictionnaires ´electroniques et analyse automatique de textes. Le syst`eme INTEX. Paris:Masson.

[37] Paroubek, P., Adda, G., Mariani, J. & Rajman, M. (1997). Les procédures de mesure automatique de l’action GRACE pour l’ évaluation des assignateurs de Parties du Discours pour le Français. In Actes Journ´ees Scientifiques et Techniques du R´eseau Francophone de l’Ing´eni´erie de la Langue de l’AUPELF-UREF (pp. 245–252). Avignon.

[38] Armstrong, S. 1996, Multext: Multilingual Text Tools and Corpora. In Lexikon und Text, pages 107-119, 1996.

[39] Ion, R., 2007. Word Sense Disambiguation Methods Applied to English and Romanian. PhD thesis (in Romanian). Romanian Academy, Bucharest.

[40] Frunza, O; Inkpen, D. and Nadeau, D. (2005) A Text Processing Tool for the Romanian Language. Proc. of the EuroLAN 2005 Workshop on Cross-Language Knowledge Induction.

[41] D. Tufiş, R. Ion, A. Ceauşu, D. Ştefănescu, "RACAI's Linguistic Web Services", in Proceedings of the 6th Language Resources and Evaluation Conference – LREC 2008, Marrakech, Morocco, May 2008, ELRA.

[42] Dimitrova, L., Erjavec T., Ide, N., Kaalep, H. J., Oravetz, C., Petkevic, V., Tufiş, D. Multext-East: Overview of the project. În Proceedings of the ALLC-ACH 98 Conference, Debrecen, Hungary, July 5-10 1998.

[43] Erjavec, T. MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 04, Lisbon, Portugal, 2004.

[44] Tufiş, D., Barbu, A. M. Specifications and Notation for MULTEXT-East Lexicon Encoding, Chapter 5, 2001, Multext-East, Concede, TELRI EU projects.

Authors

Marius Zubac (b. June 13, 1957) received his M. Sc. in Electrical Engineering (1982) from Technical University of Cluj-Napoca, Romania. Now he is a researcher and PhD student within the Computer Science Department of the Automation and Computers Faculty, Technical University of Cluj-Napoca. His current research interests include modelling of complex technical processes modelling and also natural language processing (NLP). He has authored two books, more than 10 papers, and designed complex software systems both in Romania and Canada.

Vasile-Teodor Dădârlat (b. January 13, 1955) received his M.Sc. in Computer Science (1980) from „Politehnica" University Bucharest and PhD in Computer Science (1995) from the Technical University of Cluj-Napoca. Now he is full professor of computer sciences within the Computer Science Department of the Automation and Computers Faculty, Technical University of Cluj-Napoca, Romania and director of associated „Computer Networks" Research Lab. His current research interests include different aspects of communications networks and protocols, digital circuits and e-learning systems. He has (co)authored 15 books and more than 60 papers, has more than 40 conferences participation, and has served on the TPCs of major conferences in networking and e-learning. Prof. Dadarlat has received a number of awards from the Romanian academic and technical bodies.