Formal Methods of Tokenization for Part-of-Speech Tagging[⋆]

Jorge Graña, Fco. Mario Barcala, and Jesús Vilares

Departamento de Computación
Facultad de Informática
Universidad de La Coruña
Campus de Elviña s/n
15071 - La Coruña
Spain
{grana, barcala, jvilares}@dc.fi.udc.es

Abstract. One of the most important prior tasks for robust part-of-speech tagging is the correct tokenization or segmentation of the texts. This task can involve processes which are much more complex than the simple identification of the different sentences in the text and each of their individual components, but it is often obviated in many current applications.
Nevertheless, this preprocessing step is an indispensable task in practice, and it is particularly difficult to tackle it with scientific precision without falling repeatedly in the analysis of the specific casuistry of every phenomenon detected.
In this work, we have developed a scheme of preprocessing oriented towards the disambiguation and robust tagging of Galician. Nevertheless, it is a proposal of a general architecture that can be applied to other languages, such as Spanish, with very slight modifications.

1 Introduction

Current taggers assume that input texts are already tokenized, i.e. correctly segmented in *tokens* or high level information units that identify every individual component of the texts. This working hypothesis is not realistic due to the heterogeneous nature of the application texts and their sources.

Some languages, like Galician or Spanish, show phenomena that we have to handle before tagging. Among other tasks, the segmentation process takes charge of the identification of information units such as sentences or words. This process can be more complex than it may seem *a priori*. For instance, the identification of sentences is usually performed by considering certain punctuation marks. However, a simple dot can indicate the end of a sentence, but it could also correspond to the end of an abbreviation.

[⋆] This work has been partially supported by the European Union (under FEDER project 1FD97-0047-C04-02), by the Spanish Government (under project TIC2000-0370-C02-01), and by the Galician Government (under project PGIDT99XI10502B).

In the case of words, the problem is that the spelling of word does not always coincide with the linguistic concept. Therefore, we have two options:

1. The simpler approaches just consider "spelled words" and extend the tags in order to represent relevant phenomena. For instance, the Spanish word `reconocerse` (*to recognize oneself*) could be tagged as `V000f0PE1`[1] even when it is formed by a verb and an enclitic pronoun, and the words of the Spanish expression `a pesar de` (*in spite of*) would be respectively tagged as `C31`, `C32` and `C33` even when they constitute only one term. However, this approach is not valid for Galician because its great morphological complexity would produce an excessive growth of the tag set.
2. Another solution is not to extend the basic tag set. As advantages, the complexity of the tagging process is not affected by a high number of tags, and the information relating to every linguistic term can be expressed more precisely. For instance, values of person, number, case, etc., can now be assigned to what was a simple pronoun before. As a drawback, this approach makes the tasks of the tokenizer more complex. Now, it not only has to identify "spelled words", but often also has either to split one word into several words, or join several words in only one.
   The greatest troubles arise when this segmentation is ambiguous. For instance, the words in the Spanish expression `sin embargo` will normally be tagged together as a conjunction (*however*), but in some context they could be a sequence of a preposition and a noun (*without seizure*). In the same way, the Spanish word `ténselo` can be a verbal form of `tener` with two enclitic pronouns (*hold it for him, her or them*), or a verbal form of `tensar` with only one pronoun (*tauten it*). This phenomenon is very common in Galician, not only with enclitic pronouns, but also with some expressions. For instance, the Galician word `polo` can be a noun (*chicken*), or the contraction of the preposition `por` and the article `o` (*by the*), or even the verbal form `pos` with the enclitic pronoun `o` (*put it*).

In our work, we have chosen the second option, i.e. to split and to join (to split e.g. the verb and their pronouns, and to join e.g. the different constituents of an expression). The first option, i.e. to work at the level of "spelled words", would in any case need a postprocessing step after tagging in order to identify the different syntactic terms of a text. This postprocessing step would perform tasks analogous to the ones involved in our preprocessor.

In this way, the aim of the present work is to develop a modular preprocessor, with generic algorithms, that can be used for different languages, but with better performance when linguistic information related to a particular language is provided. Therefore, it is also important to define what type of linguistic information is useful and how it will be integrated in the system in the cases where it is available.

[1] The tags that appear in this work come from projects GALENA (*Generation of Natural Language Analyzers*) and CORGA (*Reference Corpus of Current Galician*). Ap. A shows the description of every used tag. See `http://coleweb.dc.fi.udc.es` for more information of both projects.

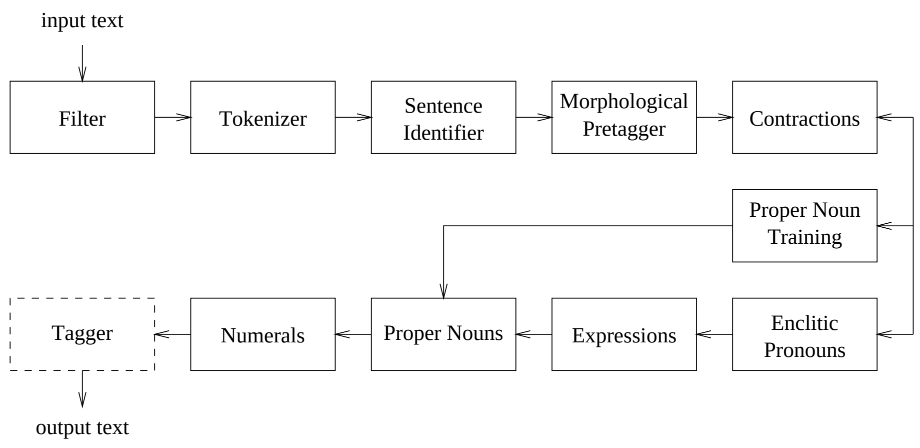

Fig. 1. General architecture of the preprocessor

Moreover, our preprocessor is specially designed as a prior phase of tagging, and it will also perform pretagging tasks. The underlying idea consists of letting the module that has the most information about a given phenomenon disambiguate this phenomenon. Therefore, as a second objective, we will also give the theoretical description of our tagger in order to complete the global presentation of the whole disambiguation process.

2 General Architecture of the Preprocessor

This section describes the different modules that are present in our preprocessor. These modules are shown in Fig. 1.

Filter. This module compacts delimiters (e.g. it removes multiple blanks or blanks at beginning of sentences) and performs conversions from typical source formats (e.g. HTML or XML) to plain text.

Tokenizer. The main function of this module is to identify and separate the tokens present in the text, in such a way that every individual word as well as every punctuation mark will be a different token. The module considers abbreviations, acronyms, numbers with decimals, or dates in numerical format, in order not to separate the dot, the comma or the slash (respectively) from the preceding and/or following elements. For this purpose, it uses two dictionaries (one of abbreviations and another one of acronyms), and a small set of rules to detect numbers and dates.

Sentence identifier. This module identifies sentences [3, 5, 6]. This task is more complex than it may seem *a priori*. The general rule consists of separating a sentence when there is a dot followed by a capital letter. However, we must take into account certain abbreviations to avoid marking the end of a sentence at their dots. For instance, this is the case of `Sr. González` (*Mr. González*). The module also considers acronyms so as not to separate their individual capital letters.

Morphological Pretagger. The function of this module is to tag elements whose tag can be deduced from the morphology of the word, and there is no more reliable way to do it. In this way, numbers are tagged with `Cifra`, and the tag `Data` is assigned to dates in formats like `7/4/82` or `7 de abril de 1982` (*April 7th, 1982*). In this latter case, we use the symbol `&` to join the different elements of the token, as can be seen in the following output:

```
7&de&abril&de&1982 [Data 7&de&abril&de&1982]
```

where the items inside the square brackets correspond to the tag and the lemma of the token under consideration.

Contractions. This module splits a contraction into their different tokens. At the same time, it assigns a tag to every one of them, by using external information on how contractions are decomposed. The module can work over other languages just by changing this information. For instance, the corresponding output for the Galician contraction `do` (*of the*) is:

```
de [P de]
+o [Ddms o]
```

i.e. `do` has been decomposed into the preposition `de` and the article `+o`. Note that the symbol `+` shows that an excision has taken place.

Proper Noun Training. Following [4–6], this module identifies the words that begin with a capital letter and appear in non-ambiguous positions, i.e. in positions where if a word begins with a capital letter then it is a proper noun. For instance, words appearing after a dot are not considered, and words in the middle of the text are considered. These words are added to a dictionary which is used later by the module Proper Nouns.

Enclitic pronouns. This module analyses the enclitic pronouns that appear in verbal forms. This is a major problem in Galician, where we can find up to four or five pronouns joined to the verbal form. The objective is to separate the verb from its pronouns and tag every one of them correctly. In order to perform this function, this module uses the following:

- A dictionary with as many verbal forms as possible.
- A dictionary containing the greatest possible number of verbal stems capable of presenting enclitic pronouns.
- A list with all the valid combinations of enclitic pronouns.
- A list with the whole set of enclitic pronouns, together with their tags and lemmas.

For instance, the decomposition of the Galician word `comelo` (*to eat it*) is:

```
come [V0f000 comer]
     [V0f1s0 comer]
     [V0f3s0 comer]
     [Vfs1s0 comer]
     [Vfs3s0 comer]
+o   [Raa3ms o]
```

where we can see that the components are `comer` (which can be infinitive, conjugated infinitive or subjunctive future) and `+o` (which is the pronoun).

Expressions. This module joins together the different tokens that make up an expression [2]. It uses two dictionaries: the first one with the expressions that are uniquely expressions, e.g. `a pesar de` (*in spite of*), and the second one with those that may be expressions or not, e.g. `sin embargo` (*however* or *without seizure*). In this case, the preprocessor simply generates all the possible segmentations, and then the tagger selects one of those alternatives later. The formalism used by our preprocessor to represent this kind of phenomenon has the following aspect:

```
<alternative>
   <alternative1>
      sin
      embargo
   </alternative1>
   <alternative2>
      sin&embargo
   </alternative2>
</alternative>
```

In Sect. 4 we will explain further how the tagger considers this representation in order to perform the disambiguation properly.

Proper Nouns. This module uses a specific dictionary of proper nouns to which proper nouns identified by the Proper Noun Training module can be added, as we saw above. With this resource, this phase of the preprocessor is able to detect proper nouns whether simple or compound, and either appearing in ambiguous positions or in non-ambiguous ones.

Numerals. This module joins together several numerals in order to build a compound numeral. For instance, every component of `mil ciento veinticinco` (*one thousand one hundred and twenty-five*) is joined with the rest in the same way as the components of an expression, obtaining only one token. Unlike the case of expressions, the tag assigned by the preprocessor here is definitive.

3 Mixed Problems

In order to form an impression of the complexity of the problems detected, we give some examples of typical cases that were solved.

Example 1. Consider the Galician expression `polo tanto`. It is an uncertain expression, i.e. `polo tanto` can be an expression (*therefore*); in its turn, `polo` can be a noun (*chicken*), a contraction (*by the*) or a verb with an enclitic pronoun (*put it*); and on the other hand, `tanto` can be a noun (*goal*) or an adverb (*so much* or *both*), when it does not form part of the expression. The preprocessor represents all the alternatives as follows:

```
<alternative>
   <alternative1>
      polo [Scms polo]
      tanto
   </alternative1>
   <alternative2>
      por  [P por]
      +o   [Ddms o]
      tanto
   </alternative2>
   <alternative3>
      po   [Vpi2s0 pór] [Vpi2s0 poñer]
      +o   [Raa3ms o]
      tanto
   </alternative3>
   <alternative4>
      por&+o&tanto
   </alternative4>
</alternative>
```

The following set of sentences contains examples of every different sense:

- **Noun+Adverb**:
  `Coméche-lo polo tanto, que non quedaron nin os osos`
  (*You chewed the chicken so much that not even the bones are left*).
- **Preprosition+Article+Noun**:
  `Gañaron o partido polo tanto da estrela do equipo`
  (*They won the match by the goal of the star of the team*).
- **Verb+Pronoun+Adverb**:
  `Pois agora, polo tanto ti coma el`
  (*So now, both you and he should put it*).
- **Expression**:
  `Estou enfermo, polo tanto quédome na casa`
  (*I am ill, therefore I am staying at home*).

Example 2. As we saw before, an example of conflict between two possible decompositions of enclitic pronouns is the Spanish word `ténselo`, which can be `tense` plus `lo` (*tauten it*), or `ten` plus `se` plus `lo` (*hold it for him, her or them*), yielding these two alternatives:

```
<alternative>
   <alternative1>
      ténse  [V2spm0 tensar]
      +lo    [Re3sam el]
   </alternative1>
   <alternative2>
      tén    [V2spm0 tener]
      +se    [Re3yyy se]
      +lo    [Re3sam el]
   </alternative2>
</alternative>
```

4 The Tagger

Although the presentation of our tagger is not one of our main objectives, a short description of its working principles is of certain importance. Due to the ambiguous segmentations described above, this tagger must be able to deal with streams of tokens of different lengths. That is, it not only has to decide the tag to be assigned to every token, but also to decide whether some of them form or not the same term, and assign the appropriate number of tags on the basis of the alternatives provided by the preprocessor. For instance, we show in Fig. 2 the streams to be evaluated if the third word has four possible segmentations.

To perform this process, we could consider the individual evaluation of every trellis and their subsequent comparison, in order to select the most probable one. It would therefore also be necessary to define some objective criterion for that comparison. If the tagging paradigm used is the framework of the hidden Markov models [1], as is our case, that criterion could be the comparison of the normalization of the cumulative probabilities[2]. One reason to support the use of hidden Markov models is that, in other tagging paradigms, the criteria for comparison may not be so easy to identify.

Be that as it may, the individual evaluation of every possible combination of alternatives could involve a very high computational cost. For instance, if another word with two possible segmentations appears in the same sentence, we would have $4 \times 2 = 8$ different streams of tokens. For this reason, we prefer to design an extension of the Viterbi algorithm, able to evaluate streams of tokens

[2] Let us call p_i the cumulative probability of the best path (the path marked with the thickest line) in the trellis i of Fig. 2. These values, i.e. p_1, p_2, p_3 and p_4, are not directly comparable. But if we use logarithmic probabilities, we can obtain normalized values by dividing them by the number of tokens. In this case, $p_1/5$, $p_2/6$, $p_3/7$ and $p_4/7$ are now comparable.

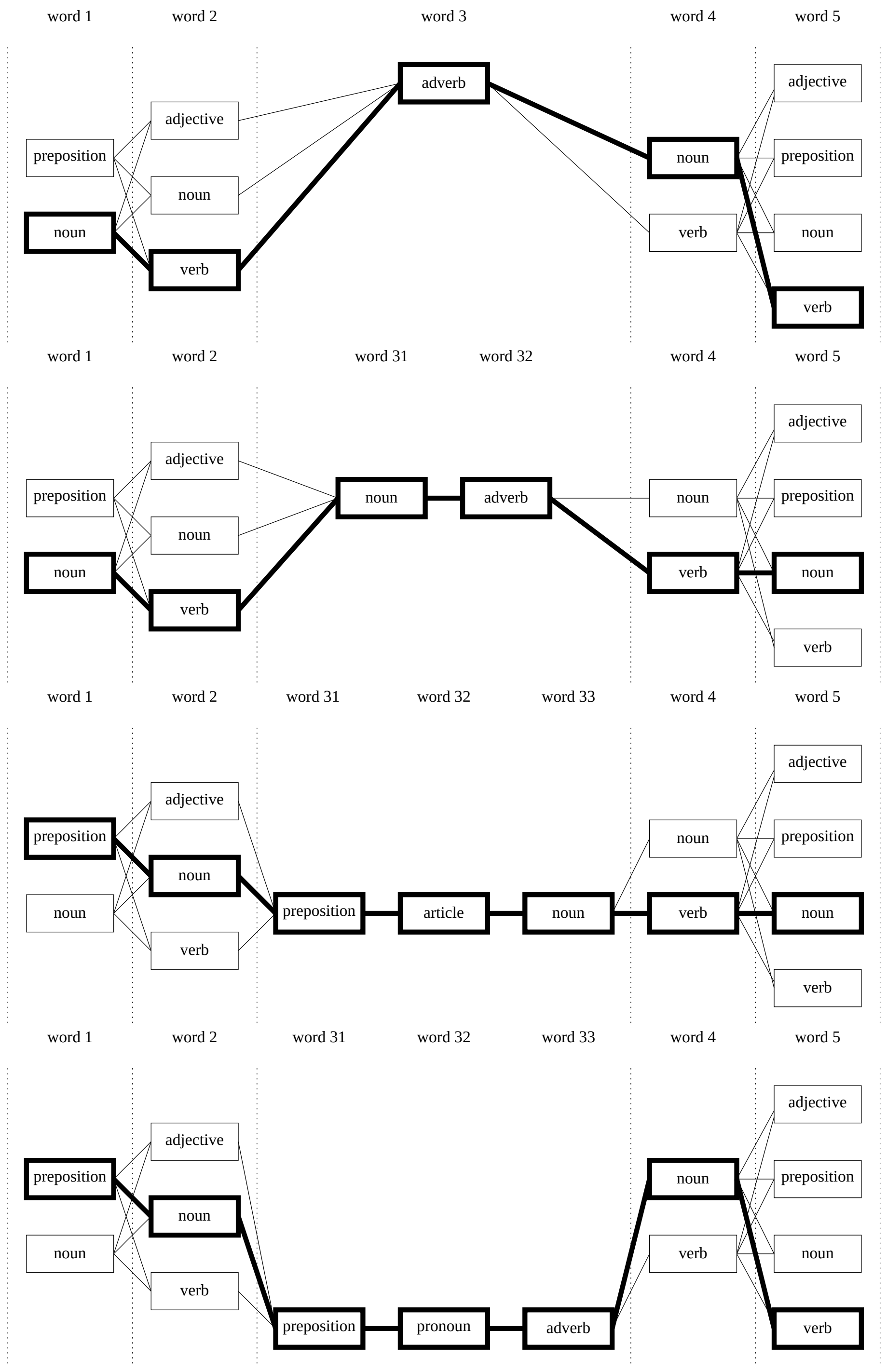

Fig. 2. Set of trellises for a set of different segmentations

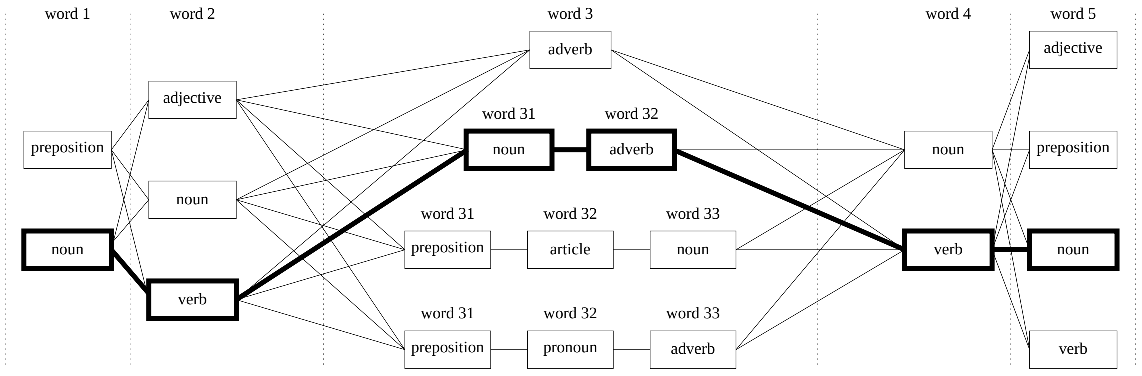

Fig. 3. A set of different segmentations represented in the same trellis

of different lengths over the same trellis (see Fig. 3) with a time complexity comparable with that of the classic algorithm. This dynamic extension is still an item of future work, but it will constitute the final step and its output will be precisely the now segmented and disambiguated text.

5 Conclusion

This work is focused on the description of specific and formal methods for preprocessing and tokenization. As we have shown, the complexity of the phenomena that appear at this level is so high that there does not even exist a strategy to determine the correct order of handling these phenomena. Our proposal attempts to fill this gap by a general scheme that avoids the particular casuistry of every phenomenon detected and every language.

The explanation has been oriented towards obtaining improvements in tagging. Nevertheless, tokenization is not only useful for automatic disambiguation. The place of the tagger could be filled by any other kind of analyser (syntactic, semantic, etc.), or simply by a *scanner* which provides all the possible segmentations and their corresponding tags, allowing the tasks involved in the manual process of building new reference texts to be performed more comfortably. In fact, this latter use is currently being intensely exploited for Galician, a language for which linguistic resources hardly exist.

Among our future proposals, besides the implementation of a dynamic version of the Viterbi algorithm, we aim to improve the generalization of our algorithms to simplify their adaptation to other languages. On the other hand, we also need to cover more and more preprocessing tasks that are still under study, but without using a great amount of linguistic resources that may not exist. If they are available, they would be used simply to refine the global behaviour.

The final objective of the general architecture of preprocessing presented here is its integration in an information retrieval system, and we expect that the modules described will contribute to improving the performance of the system.

References

1. Brants, T. (2000). TNT - A statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP'2000)*, Seattle.
2. Chanod, J.-P.; Tapanainen, P. (1996). A Non-deterministic Tokeniser for Finite-State Parsing. In *Proceedings of the Workshop on Extended finite state models of language (ECAI'96)*, Budapest.
3. Grefenstette, G.; Tapanainen, P. (1994). What is a word, What is a sentence? Problems of Tokenization. In *Proceedings of 3rd Conference on Computational Lexicongraphy and Text Research (COMPLEX'94)*, July 7-10.
4. Mikev, A. (1999). A knowledge-free Method for Capitalized Word Disambiguation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, June 20-26, Maryland.
5. Mikev, A. (2000). Document Centered Approach to Text Normalization. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2000)*, July 24-28, Athens, pp. 136-143.
6. Mikev, A. (2000). Tagging Sentence Boundaries. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'2000)*, Seatle, pp. 264-271

A Description of the tags

This appendix describes the tags that appear in the examples. As we mentioned above, these tags come from the tag sets of the projects GALENA and CORGA.

GALENA tags	
`C31`	First element of a conjunction of three elements
`C32`	Second element of a conjunction of three elements
`C33`	Third element of a conjunction of three elements
`Re3sam`	Pronoun: enclitic, accusative, masculine, third person singular
`Re3yyy`	Pronoun: enclitic, accusative or dative, masculine o feminine, third person, singular or plural
`V000f0PE1`	Verb: infinitive with one enclitic pronoun
`V2spm0`	Verb: present, imperative, second person singular

CORGA tags	
`Cifra`	Number
`Data`	Date
`Ddms`	Article: determinant, masculine, singular
`P`	Preposition
`Raa3ms`	Pronoun: atonic, accusative, masculine, third person singular
`Scms`	Noun: common, masculine, singular
`V0f000`	Verb: infinitive
`V0f1s0`	Verb: infinitive conjugated, first person singular
`V0f3s0`	Verb: infinitive conjugated, third person singular
`Vfs1s0`	Verb: future, subjunctive, first person singular
`Vfs3s0`	Verb: future, subjunctive, third person singular
`Vpi2s0`	Verb: present, indicative, second person singular