EtiFac: A facilitating tool for manual tagging[1]

António Horta Branco
João Ricardo Silva
Faculty of Sciences, University of Lisbon

1. Introduction

The pervasiveness of ambiguity is a major issue in the parsing of natural languages. In this respect, morphosyntactic tagging can be seen as a first step in the process of resolving ambiguity, a step that proceeds only on the basis of morphologic and very local syntactic information.

Although automatic morphosyntactic tagging does not deliver the depth of analysis of a full blown grammatical parser, it turns out to be very important from a practical point of view. On the one hand, the decision about the categorial tag to be assigned to a given token can be done by resorting to procedures which are computationally much less expensive than what is required by context-free parsing algorithms. On the other hand, the removing of categorial ambiguity at a preliminary stage of the parsing process represents a crucial contribution to alleviate the workload required from the subsequent, deep syntactic and semantic, parsing steps.

In recent years, tagging based on statistical methods has been developed, representing an important contribution to enhancing the performance of automatic taggers. This type of techniques can deliver very good results, with an accuracy up to 95%-97% [Manning and Schütze, 1999, p.371]. A major difficulty for using these techniques, however, has to do with the fact that they require a previously tagged corpora on which to perform the statistical training.

In order to obtain these correctly tagged corpora, human manual tagging is inevitably called for, either for directly tagging each token, or to check and correct the tags assigned by an initial automatic tagger in the process of being perfected. The crucial point to underline here is that manually assigning tags to every token of a corpus with several million words is a highly demanding task in terms of resources since it requires the continued effort of many linguistically skilled people for a very long period of time.

In this connection, aiming at improving the speed and accuracy of manually tagging corpora, we developed **EtiFac**, a tool for supporting and facilitating the operations involved in such a task. In this paper, we will report on this tool and on how it can be used to help on the tagging process.

[1] The results described in this paper were developed in the project NEXING-Natural Negation Modeling and Processing. This project is supported by FCT-Fundação para a Ciência e Tecnologia under the contract FCT/SAPIENS99/34076/99. It can be visited at http://www.di.fc.ul.pt/~ahb/nexing.htm.

***Actas do XVII Encontro Nacional da Associação Portuguesa de Linguística*, Lisboa, APL, 2002, pp. 81-89**

In Section 2, we will discuss the first component of **EtiFac**, a morphosyntactic tagging tool which basically handles the so called categorial closed classes.

In Section 3, we will introduce the second component of **EtiFac**, an editing tool that helps the user to do two things: one is to tag the tokens that the first component did not tag, i.e. tokens of the open classes; the other is to choose one of the many tags possibly assigned to a given token by the automatic tagger.

In Section 4, we will present the developments of **EtiFac** we are working in at present.

2. First component: the tagger

The first component of **EtiFac** tags the tokens belonging to closed syntactic classes and the tokens of the open syntactic classes exhibiting productive derivational morphology.

2.1 Closed syntactic classes

With this component, we seek to take advantage of the fact that the few hundreds of items belonging to closed classes are typically highly frequent, thus covering a very large percentage of the corpus. For example, when tagging a sample of 10% of the CETEMPúblico corpus only with tags of closed classes, we observed that it is possible to cover nearly two thirds of that sample.

Most of these items can be tagged with complete certainty and in a deterministic way. In order to achieve this, we make our tagger to scan every token in the text at stake. Each token is checked out against a list containing the items of the closed classes and, if a given token is present in that list, it is assigned the tag of the corresponding category. When there is ambiguity, i.e. when a token may be assigned more than one tag, we use portmanteau tags. This type of tags represents all the possible categories for a given token, including tags of open class categories, thus leaving disambiguation for a subsequent stage of processing, when the disambiguation process may count on contextual information already detected during this first tagging pass.

A considerable advantage of this first level of automatic tagging is that it can be done very rapidly. To find the tag or tags for a given token, **EtiFac**'s first component needs only to check for the occurrence of that token in a list. Another advantage has to do with the fact that the lists can be kept independent of the tagger engine. Simply by changing these lists we can then obtain a different tag set, a different list of lexemes for a given category, or even a different tagger for another language.

This tagger recognizes both single- and multi-word expressions. Multiword units (MWU) are tagged using the longest match criterion. For example, the MWU *de forma a que* will be chosen for tagging, even though *de forma a* is also a MWU

available in the closed classes list. When a sequence of tokens may be analyzed both as MWU and as a sequence of simpler expressions, the representation of the MWU in the closed classes files encodes all such options, as in the following example: `_visto que_CJ//visto_PTP que_QUE//visto_CN que_QUE`.

As to contractions, such as *disso*, *à*, *daquele*, etc., the component elements are separated and tagged. Tagging the elements involved in contractions may allow already to solve some ambiguous cases. Some items, when alone, can be classified in several ways, but when they are part of a contraction one knows exactly which tag should be used. For example, `a` is tagged as `a_DA_PREP_CL` by default, but when the contraction pela is found, the resulting tagging will be `a_DA`, indicating that, in this occurrence, this token is a definite article.

After a contraction has been expanded, it may happen that its first element can be accepted as part of a MWU together with the items preceding that contraction. For instance, in *junto do*, the contracted *do* expands to *de o*, thereby originating the sequence *junto de o*, which then permits the analysis of the preposition *junto de* as a MWU. Accordingly, **EtiFac** ensures that, if the action of breaking the contraction allows the formation of a MWU, that MWU will be correctly tagged.

The tagger also separates clitic pronouns from verbs when the former are found either in enclisis or in mesoclisis. By looking into the list of the clitics, it is quite easy to separate the clitics away from the adjacent word thus allowing the tagging of this word as a verb. This way, even though we were concerned basically with the closed classes, we already manage to tag some verbs, namely those occurring together with a clitic. For example, *juntar-se* is tagged as `juntar_V se_CL`. The particular case of *haver-de*, in which *de* is tagged as a preposition, is also handled by the same strategy used for clitics.

As for punctuation symbols, the tagger marks the existence of adjoining blank spaces to a punctuation mark by adding an asterisk to the left and/or right of it. In a latter stage, this allows one to take a more informed decision about the specific role of that punctuation symbol in the occurrence at stake. For example, the decimal point in 3.5 is tagged simply as `._PNT`, but a period ending a sentence and appearing with a blank space to its right is tagged as `.*_PNT`.

2.2 Open syntactic classes

The tagger also copes with some items belonging to open syntactic classes. For this purpose, it basically resorts to the same strategy of checking tokens found in the text being scanned against a list of items previously stored in memory. When dealing with closed classes, these items in the list are the expressions to be tagged. As for open classes, what is kept in a memory is a list of terminations of words that, inasmuch as they may result from a productive process of morphological derivation, they can be taken as indicators of the category (Noun, Adjective, etc.) of the corresponding full word. For instance, words ending in *-são* are tagged as Nouns.

This strategy allows the tagger to handle a considerable number of tokens without resorting to a lexicon where words of the open classes would be stored together with the information about their categories. Nevertheless, this strategy has also to take into account that there may exist many exceptions to such generalizations. For instance, although the great majority of words ending in *-são* are nouns, some of them, like `geresão_ADJ` or `malsão_ADJ`, are not. Given that the exceptions to derivational regularities exist in a manageable number, they are just kept in a list together with their correct tag: If a given word belongs to this list of exceptions, it is tagged with the category stated there; if it does not belong to it, then the general rule for the identified termination is used.

Another clue used to increase the coverage of the tagger without resorting to a lexicon is to rely on the fact that human writers implicitly tag proper nouns by capitalizing their first letter. Accordingly, a word that does not occur in the beginning of a sentence and that starts with a capital letter is tagged as a proper noun. There are however exceptions to this rule, of which social titles are the most common: *Presidente Sampaio*, *Sargento John Smith*, etc. These exceptions are listed in a file and checked out before applying the general heuristic for capitalized proper nouns.

As for the words appearing in the beginning of sentences, they are all supposed to begin with a capital letter. In this case, every item in the beginning of sentences is checked against a list of the more frequent proper nouns: If a given item is found in that list, it is tagged as a proper noun.

Roman numerals are also recognized and they are tagged with `_DGTR`, standing for "digit: roman". There is a list of items that happen to be ambiguous between roman numerals and other words. These items receive all possible tags. This is the case of, for example, `vi_DGTR_V`, `li_DGTR_V`, etc.

Given their very specific format, sequences of tokens corresponding to URL and email addresses are recognized and tagged with `_EADR`.

2.3 Technical matters

The tagger has been developed using the C programming language in the Linux platform, and so far it was tested only in this operating system. Given it was implemented in C, it should compile and run easily in any platform as long as the `flex` libraries are made available.

The tagger is run from the command line. There are nine arguments that should be used in the command line, each of them referring a text file:

- `-f` corpus to be tagged
- `-c` list of lexemes from closed classes
- `-t` list of contracted forms
- `-l` list of clitics and related endings
- `-s` list of terminations

-e list of exceptions to derivation rules
-n list of exceptions to the heuristic for proper nouns
-p list of frequent proper nouns
-o list of "odd" sequences (currently: ambiguous roman numerals)

If the tagger is run with no options defined or with incorrect arguments, a short help message is displayed to remind the user of the required options.

The files are in plain text and have to comply with a specific format:

-c **File with closed classes forms** This is a file separated in sections. The beginning of each section is marked by the heading _X, where X is the tag that will be given to the lexemes below that heading. The file contains one non-indented item per line. A line with comments is prefixed with %%. Example:

```
...
_PREP
a
ante
%% This is a comment
_CJ
apesar de
mas
...
```

-t **File with contracted forms** This is a file where each entry is a three line block. The first line is the contracted form and is ended by a comma. The second line is the expanded form of the contraction and is ended by a comma. The third line is the expanded form of the contraction tagged and ended by a period. Example:

```
...
nesta,
em esta,
em_PREP esta_PRON.
neste,
em este,
em_PREP este_PRON.
...
```

-l **File with clitics and related endings** This is a file with a clitic, a contraction of clitics or a "related" ending per line. This list is formatted like the list for closed classes, with a _X heading preceding the group of items with that tag. Contracted clitics are displayed in a line with the contracted form, followed by a comma, followed by the corresponding expanded forms, which are associated with the relevant tag. The verb endings involved in mesoclisis are preceded by /. Example:

```
...
_CL
me
te
...
mo,me_CL o_CL
mos,me_CL os_CL
...
/ei
/ás
...
```

`-s` **File with terminations** Text file with the same format as the file with the list of items from closed classes. Each section heading indicates the tag to be attributed to the lexemes exhibiting the terminations listed below that heading. Example:

```
...
_ADV
mente
_CN
agem
são
...
```

`-e` **File with exceptions to derivation rules** Text file with the same format as the file with the list of items from closed classes. This file contains the items that exhibit the terminations specified in the file described just above but that do not have the categories that, as a rule, are expected for the items with those terminations. Example:

```
...
_ADJ
clemente
comprimente
deprimente
...
_CN_ADJ
selvagem
...
_ADJ
geresão
...
```

`-n` **File with exceptions to the heuristic for proper nouns** Text file with the same format as the file with the list of items from closed classes. Example:

```
_STT
presidente
ministro
papa
cardeal
bispo
...
```

The lists of items in the files described above are read from these files and stored internally as balanced binary trees. This is a standard technique used to reduce the number of steps needed to find an item in a list. While searching sequentially through an ordered list of n items has complexity $O(n/2)$, finding an item in a balanced binary tree is only of complexity $O(\log_2 n)$, which for large lists implies a searching time several orders of magnitude smaller. A possible disadvantage of using balanced binary trees is the overhead required for inserting or deleting items during execution. This is not, however, a problem in the present case since here the trees are created at the activation of the tagger and are not changed during its execution.

3. Second component: the manual tagging assistant

After using the tagger described above over a corpus, there will be items remaining to be tagged. At this point, however, an important progress was already made with regards the final goal of obtaining a corpus completely and accurately tagged.

On the one hand, the number of occurrences of untagged items in the corpus is now much smaller, around 1/3 of the total number of tokens in the corpus to be tagged (cf. Section 1 above). This reduces the extent of the data left to be manually processed by the human tagger.

On the other hand, given the way the tagger is designed, these yet untagged items must belong to one of the few open morphosyntactic classes. This reduces the complexity of the task for the human tagger, which has now simply to choose one tag to be assigned out of a list of three categories.

Although the length and complexity of the task has been substantially reduced, there is still a lot of effort asked for the completion of the task of fully and accurately tagging the corpus. In order to further minimize this effort, the task of manual tagging can be further assisted by the computer. This brings us to the second component of **EtiFac**, the manual tagging assistant.

For implementing this assistant, we simply used a text editor that allowed the definition of macros. Macros were used to simplify the tagging task inasmuch as sequences of actions can be grouped and triggered with a single command by the user.

One of the macros is used to jump to the next untagged item and display a pop-up menu with the possible options for tagging that item. At this point, the user needs only to select the correct option in that menu (see Annex for an example).

The other macro helps the user to search for items that have been tagged by the automatic tagger with more than one tag. Upon finding one such item, the user is then asked to choose one tag among those that were initially assigned to it.

4. Conclusions and future developments

As a tool for supporting the task of tagging, **EtiFac** is still being improved. At present we concentrate on further developing its first component. While this will permit us to obtain enhanced versions of the tagger, and in particular of **EtiFac**, our final goal is to improve that component up to a point where we eventually get a fully fledged, standalone automatic tagger.

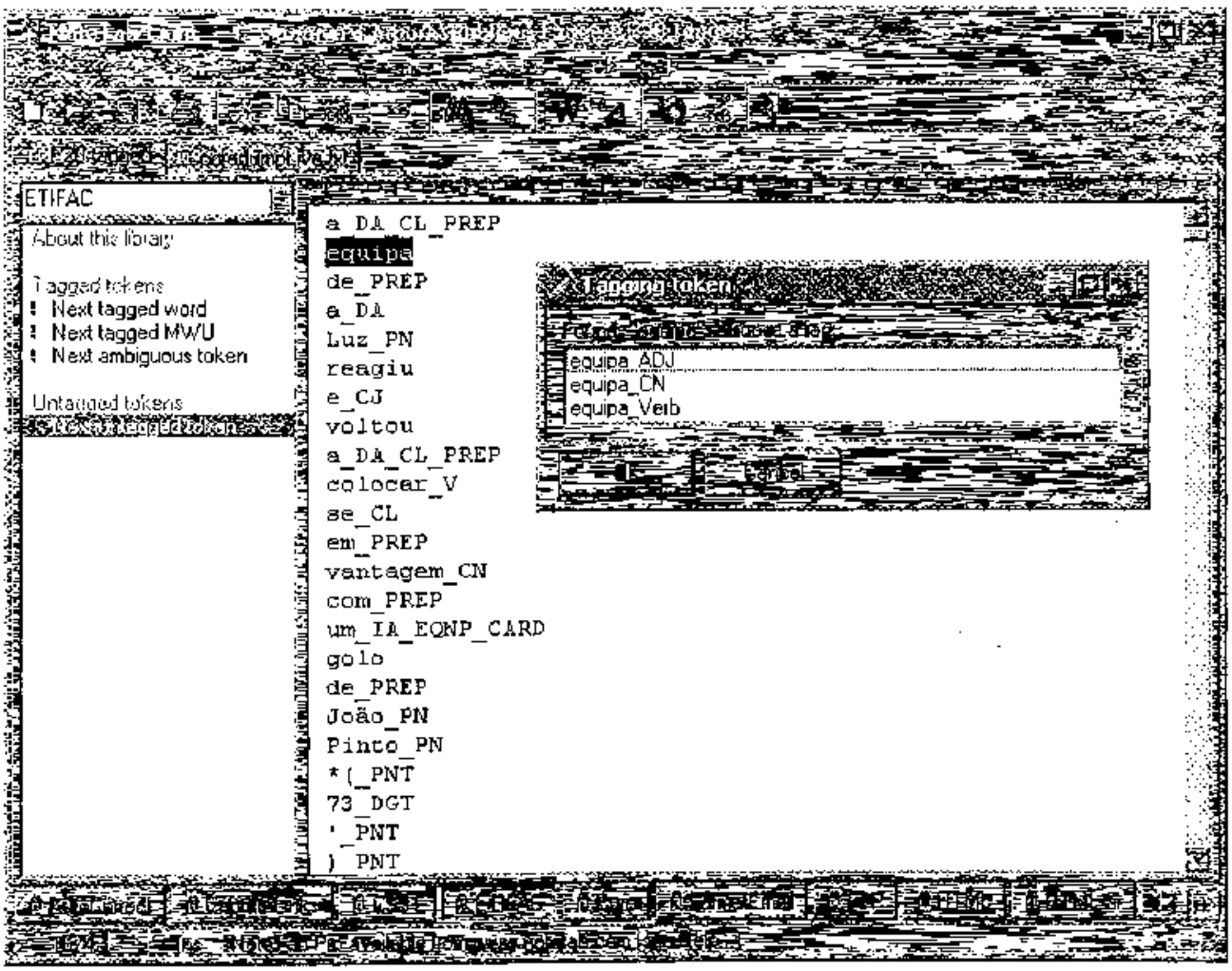
- Accordingly, one of the next tasks is to keep enlarging the lists of closed classes to ensure that they are as complete and accurate as possible.
- Prefixation is another issue to be dealt with. While we managed to implement a strategy to handle the most productive derivational terminations, there are also prefixes, such as *anti-*, *bio-*, *hidro-*, etc. that remain to be accounted for. This is important in what concerns the list of exceptions to the derivational rules. As it is not possible to list all possible items that can be formed by regular prefixation and should appear in the list, the tagger has to be prepared to automatically expand the present list by means of regular prefixation.
- The strategy used to tag items from their regular terminations was used only for derivational morphology. It can also be used to handle inflectional morphology. One of the next steps is to try to used the verbal terminations to cover as many verbal forms occurring in the corpus as possible.
- Although all these steps will allow us to improve the coverage and accuracy of the tagger, there will inevitably be a residual number of tokens left to be tagged by this deterministic process. To handle these cases, we will then have to improve the tagger developed so far by integrating statistical techniques in it.

The aspects just mentioned are essential in order to obtain a tagger with a better coverage, but there are other aspects of **EtiFac** that also ask to be improved: The command line interface to the first component of **EtiFac** is still too cumbersome. Hopefully, it would be redesigned into a configuration file with all the options.

5. References

Manning, Christopher and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

6. Annex


ETIFAC
About this library
Tagged tokens
Next tagged word
Next tagged MWU
Next ambiguous token
Untagged tokens
a_DA_CL_PREP
equipa
de_PREP
a_DA
Luz_PN
reagiu
e_CJ
voltou
a_DA_CL_PREP
colocar_V
se_CL
em_PREP
vantagem_CN
com_PREP
um_IA_EQNP_CARD
golo
de_PREP
João_PN
Pinto_PN
*(_PNT
73_DGT
'_PNT
)_PNT
Tagging token
equipa_ADJ
equipa_CN
equipa_Verb