Unknown Words Guesser

João Rebelo[1], Nuno Mamede[1,2], and Jorge Baptista[2,3]

[1] Universidade de Lisboa - Instituto Superior Técnico
[2] L^{2}F – Spoken Language Systems Laboratory – INESC ID Lisboa
[3] Universidade do Algarve - Faculdade de Ciências Humanas e Sociais
joao.humberto@tecnico.ulisboa.pt
nuno.mamede@inesc-id.pt
jbaptis@ualg.pt

Abstract. STRING is a Natural Language Processing (NLP) chain developed at L^{2}F / INESC-ID Lisboa and LexMan is the module responsible to realize the tokenization and morphological analysis. This paper addresses the challenge of improve LexMan submodule responsible for propose correct pairs (lemma + tag) to unknown words.

Keywords: Natural Language Processing · Morphological Analysis · Unknown Word Guesser · Markov Models.

1 Introduction

Natural Language Processing (NLP) is a Linguistic and Artificial Intelligent subfield that studies problems related with generation and automatic understanding of natural languages. This processing is composed by several successive stages: tokenization, morphological analysis, syntactic analysis and semantic analysis. The first step of the process is where the input text is divided by tokens, each token can be formed by one or more words, numbers, etc. After this process, the Morphological Analyzer receive and analyze these tokens to assign the morphosyntactic tags. Based on these tags, a Syntactic Analyzer performs a structured representation of the input text. Lastly, a Semantic Analyzer is responsible for returning a formal representation with possible meanings for the text.

1.1 STRING

STRING (Statistical and Rule-based Natural Language processing chain) [7]. This system has a pipeline structure and is composed by four modules: *LexMan*, *RuDriCo*, *MARv* and *XIP*.

LexMan [3] is the first module of the chain, is responsible for the input text tokenization and morphological analysis. *LexMan* processing is divided in two stages.

The dictionary generation takes place in the first stage, this process is composed by three submodules: a Words Generator, a Suffix Generator and a Transducer Generator. The first module combines a lexicon of lemmas with a set of inflection paradigms to generate all the inflected forms associated to this lemmas and to each word associate a morphosyntactic tag. The Suffix Generator [1] produces the simple and compound words generated and the derived forms by suffixation associated for each type of word. Lastly, the Transducer Generator is responsible for transducers creation for the words generated before and combines them with prefixes, clitics information and with handmade transducers. This last generated transducer corresponds to the lexicon used by *LexMan* during the morphological analysis.

During the second processing stage of *LexMan* the input text tokenization and the morphological analysis is realized. The input text transducer is composed with LexMan transducer lexicon. After that LexMan analyzer go through the tokens to assign them the obtained tags from that composition. If there are some words that had not been identified on lexicon, *LexMan* invokes *Guesser* submodule to identify this unknown words.

RuDriCo [4] is a morphosyntactic desambiguator, which is responsible to solve ambiguities not resolved by *LexMan*.To solve that, this module uses a set of segmentation and disambiguation rules. This allows join or separate two or more segments, for example `Coreia`, `do` and `Sul` into `Coreia do Sul` or `no` into `em o`.

MARv [10] is a statistical desambiguator responsible to resolve some ambiguities that may still exists and are derived from *RuDriCo*. This module is based on second order Markov Models and it uses Viterbi's algorithm [14].

XIP (Xerox Incremental Parser) [5] is responsible for text syntactic and semantic analysis. This module constructs syntactic constituents (chunks) and establishes dependency relations between them.

1.2 Context

Attending the problem of identifying unknown words in Portuguese language, there are already some tools:

- *Palavroso* [8] is a morphological analyzer that can be used as a morphological or lexical component for natural language processing systems in Portuguese. *Correcto* is one of this systems that uses *Palavroso* as a lexical component. *Correcto* [8] is a orthography corrector. This module proposes a set of alternatives to the unknown words providing processes to identify linguistic errors and typographic errors. These alternatives will be verified using *Palavroso* lexicon.

- *Palavras* [2] is a processing text system for Portuguese language in both European and Brazilian dialect. *Palmorf* [2] is the module of this system which is responsible to realize the text morphological analysis. *Palmorf* can propose alternatives for unknown words, based in European and Brazilian Portuguese differences and errors associated to incorrectly placed or missing diacritics.
- Sumit Sharma and Swadha Gupta proposed an intelligent automatic orthography corrector based on the use of Markov Models to identify spelling mistakes on text [12]. Conventional spell checkers only fix non-word errors and the real-word errors that gives valid words but are not intended by the user was not detected. In order to correct this kind of problems they propose an approach based in trigrams and Bayesian approaches.

1.3 Goal

The goal of this work is to improve the *LexMan* submodule which is responsible for proposing pairs (lemma + tag) for words that do not exist in the *LexMan* lexicon, the *Guesser*. The original solution of this submodule deals with problems related with upper cases, common errors, compound words and incorrect placed or missing diacritics. The goal of the new solution is to generate more alternatives for unknown words by using different strategies. The first task is the improvement of the original solution. Once completed, the addition of new modules can be used to handle new strategies. This new modules aims to resolve some problems reported on Alexandre Vicente [13] and Hugo Almeida [1] master's thesis.

2 Guesser Original Architecture

The *Guesser*[1] is a *LexMan* submodule, responsible for proposing pairs (lemma + tag) of words that are not in *LexMan* lexicon. This module is essentially divided into two stages. The first is executed during the tagging process and the second after all the input text has been processed.

2.1 1st Processing stage

The first stage is where *Guesser* generates the alternatives for unknown words when invoked by LexMan. This execution is divided in 5 submodules, each one can propose alternatives for the words based in different strategies. The original *Guesser* solution follows a pipeline architecture for alternatives generation by the following modules:

- the *Upper Case Converter*, the first *Guesser* submodule, is responsible to generate alternatives based on incorrect capitalization;
- the *Direct Substitution* is the module that deals with the most common typographical errors. This module saves this common errors as their corresponding tags on a file, and when the unknown word is one of this saved

errors, the module executes the respective substitution. This is the most efficient *Guesser* module because it does not need to validate an alternative word;
- the *Compounds* is a module that deals with compounded words that are not automatically generated. This module saves a set of compounded words terminations: `-alvo`, `-mestre`, `-mãe`, `-padrão` e `-chave`. If the unknown word was formed by one of this terminations, like `público-alvo`, the generated alternative will correspond to the first element of the compounded word, `público`;
- the *Diacritics* module deals with incorrect placed diacritics, and it is only invoked if the unknown word had at list one diacritic. These kind of errors are very common, such `cafe` instead of `café` or `ràpido` instead of `rápido`;
- the *Termination* module is invoked when the other modules did not produce alternatives or produced but were not in *LexMan* lexicon. This module takes into consideration the end of words. For example, words ending in `-o` are considered to be of the masculine gender, while those ending in `-a` are considered feminine. This module always assigns a tag to the unknown words, even if it is not correct.

When *LexMan* finishes processing the input text, a composition of the alternatives file is performed with the *LexMan* dictionary. The transducer obtained from that composition is used in the next processing stage.

2.2 2nd Processing stage

This second stage has the objective of validate the alternatives generated in the first stage and occurs after *LexMan* processes all the input text. Now for each unknown word, *Guesser* module will go through the transducer obtained from the composition between alternatives file and *LexMan* dictionary to verify if there is a valid alternative to the unknown word, if there is, it assigns its tag, otherwise invokes the *Termination* module to propose one.

3 Guesser Solution

This new solution comes to solve some problems identified in original architecture. Changes were made to the original architecture and modules, and five new modules were added to the *Guesser*.

3.1 Original Architecture Modifications

Before this work, the *Upper Case Converter* was the first module to be executed, and the other models could not generate alternatives for words started by upper case's. The new solution places this module on the third position of execution and allow *Guesser* to process words like `PrimeiroMinistro` or `ESTado`. Now only words in which only the first letter is uppercase are not processed, like `Blimunda`.

Other problem identified, also related with *Upper Case Converter*, was that this module always generated an alternative and verified on *LexMan* lexicon, even if the alternative had the same shape of the original word. This resulted in a significant increase in processing time. To solve that, in the new *Guesser* solution, *Upper Case Converter* will just generate an alternative if there are upper case letters in unknown word, and this alternative is added to the alternatives file as is done by other modules.

The original *Diacritic* module did not address problems related to the non-placement of `ç`'s. To solve this, the new solution save a set of possible substitutions, listed in table 1. This substitutions only occur at the end of the words, for example, for unknown `cancão`, the generated alternative will be `canção`.

original	cão	cões	ca	co	cu
substitutions	ção	ções	ça	ço	çu

Table 1. Possible substitutions of `c` to `ç`.

To optimize the execution time and not increase with the addition of the new modules. The modules *Upper Case Converter*, *Hyphen*, *Language Model*, *Phonemes* and *Diacritic* are executed in different processes to optimize the execution time, except the modules *Direct Substitution* and *Compounds*, because when these modules create an alternative, they do not requires the execution of the others.

3.2 New Modules

The following points describes the new modules added. All these modules were executed simultaneously and concurrently in different threads, and all of them generate alternatives for the same alternatives file.

Hyphen During the analysis of the identified unknown words in the corpora used in this work, the presence of some errors originated by the absence of hyphens were detected, like `PrimeiroMinistro` or `DecretoLei`. These words have characteristics that allows the identification of the presence of a uppercase letter which is neither the first nor the last. In these cases the module generates an alternative by placing a hyphen before the upper case, for example, `Primeiro-Ministro` or `Decreto-Lei`.

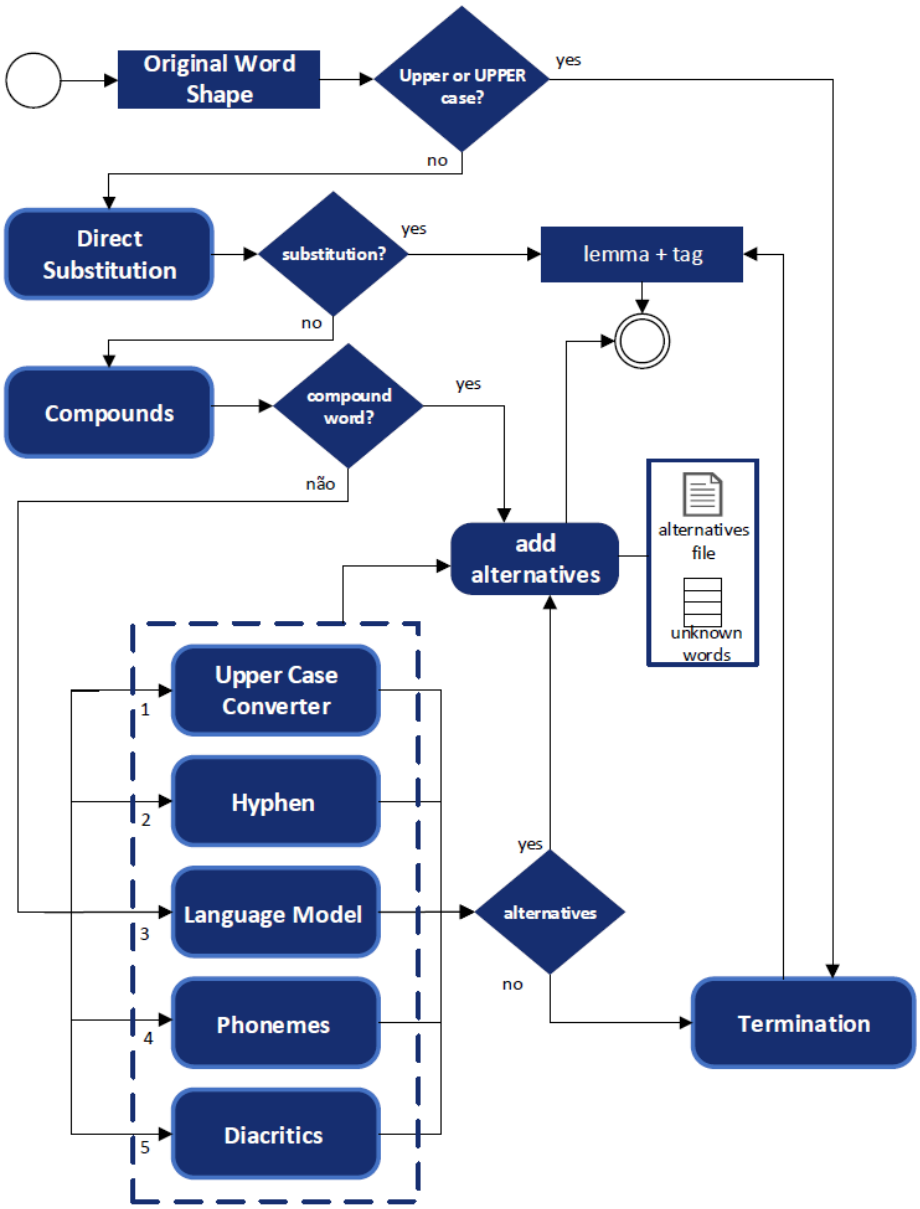

Fig. 1. New *Guesser* solution architecture - stage 1.

Phonemes This module creates alternatives based in commons phonetic errors of Portuguese language. Table 2 contains the set of substitutions that this module is allowed to do. When the unknown word has one letter or a sequence of letters presented on table 2, the module generates an alternative by replacing that for

the corresponding substitution. For example, if the unknown word was `ascenção`, the module will generate `ascensão` by replacing `ç` for `s`.

original	ç	ss	ç	s	c	ss	x	ch	z	ão	am
substitution	ss	ç	s	ç	ss	c	ch	x	s	am	ão

Table 2. Phonetic substitutions saved.

Language Model The goal of this module is to generate alternatives formed by various types of errors. To do this, the module uses a second order Markov Model and for each unknown word is also saved the two previous words within the text. With these two words, the *Language Model* search in trigrams for possible words that can appear after this sequence. In order to select the better alternative to generate, it uses the minimum editions number [6] algorithm between the alternative and the unknown word, the alternative with lower value is added to the alternatives file. The minimum number of editions is changed according to the unknown word size for there to be no alternatives that are generated that do not match the correct unknown word form.

Select Alternative This module is executed in the second processing stage of *Guesser* with the goal of selecting the best alternative when there are more than one corrected for the same unknown word. To do this, the module uses trigrams, unigrams and also the number of editions between the alternatives and the unknown word. In the original *Guesser* solution even if there was many correct alternatives, it was always chosen the first found. Now with the new implantation, is selected the most appropriated alternative.

3.3 New Guesser Architecture

As illustrated in Figure 1 1, the new architecture for the *Guesser* first stage where it is possible to observe the new position of the original modules and also the new ones. The most significant changes are the addition of 3 new modules and the parallel execution of the modules that generate alternatives to the file, except the *compounds* module.

Regarding the architecture of the *Guesser* second stage, it is presented in Figure 2. The only change made in this stage of the *Guesser* was the addition of the *Select Alternative* module.

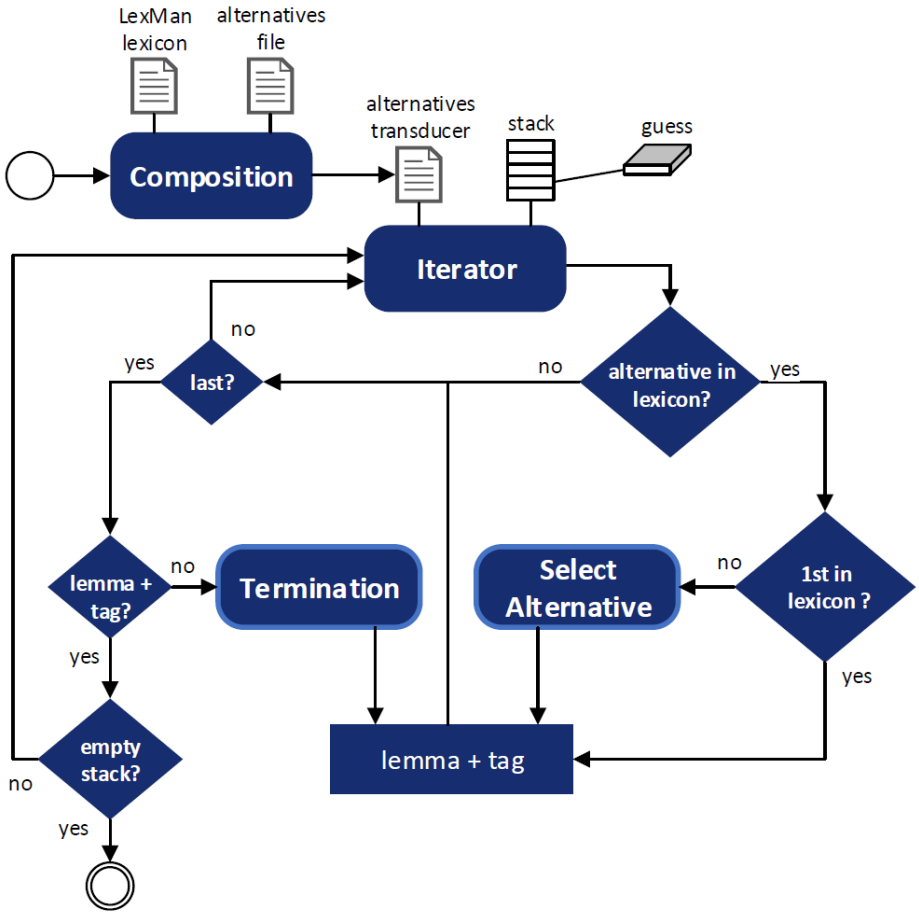

Fig. 2. New *Guesser* solution architecture - stage 2.

4 Evaluation

This section presents the results obtained in the evaluation of the new solution regarding the processing time and the precision, in the correct assignment of the pairs (lemma + tag) to the unknown words. To do this, firstly was created an excerpt with 4000 phrases which contained unknown words obtained from two corpora: one from a Portuguese newspaper, Público [11], and the other one from Portuguese Parliament [9]. In order to evaluate the correct identification of unknown words by *Guesser*, a script was created to annotate these unknown words. This annotation task was done by a Doctorate in Linguistic, a Doctorate in Electrical and Computer Engineering and a master's student in Computer Science Engineering.

4.1 Time

To evaluate the processing time, were used two approaches: one to evaluate each *Guesser* module using the excerpt created and other one too evaluate *Guesser* in a real context using different files with different sizes from Público corpus.

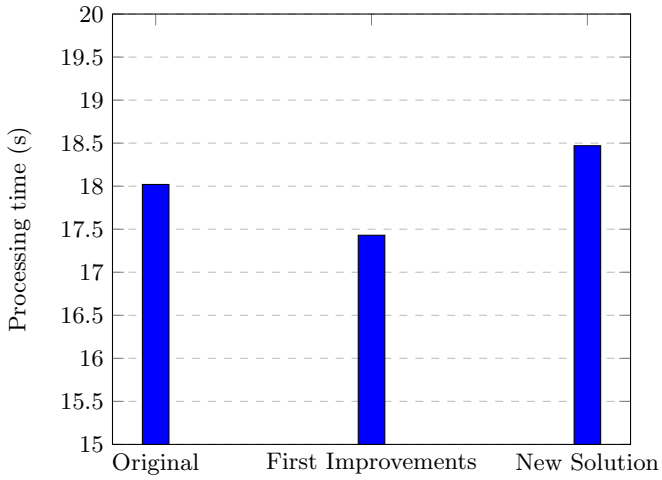

Fig. 3. Processing time results using the excerpt with 4000 sentences.

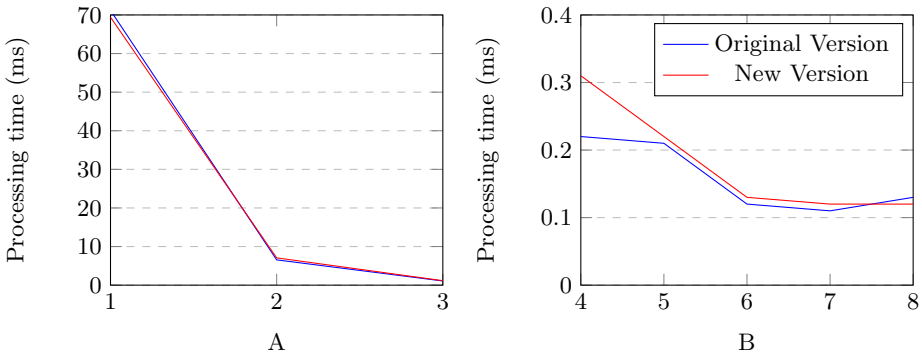

Fig. 4. Word execution time by number of sentences processed. Graph A - files 1 - 1; 2 - 10; 3 - 100 and Graph B - files 4 - 500; 5 - 1000; 6 - 5000; 7 - 10000; 8 - 50000

In the graph of Figure 3 is presented the results for time when processing the excerpt with 4000 sentences. Through the analysis of the obtained results, it is conclusive that with the improvements made to the initial architecture

the processing time decreases slightly. However, with the addition of the new modules the situation was reversed .

In the graph's of Figure 4 is presented the results for time when processing files with different number of sentences in order to simulate real context. Through the analysis of the obtained results, it is conclusive that for smaller and big files the new solution get better results then the original one.

4.2 Precision

	Off	DS	C	UC	H	LM	P	D	S	New Solution
1	✗									576 (11%)
2		✗								583 (12%)
3			✗							592 (12%)
4				✗						576 (11%)
5					✗					593 (12%)
6						✗				804 (16%)
7							✗			604 (12%)
8								✗		929 (19%)
9									✗	576 (11%)
10		✗	✗							599 (12%)
11		✗	✗	✗						599 (12%)
12		✗	✗	✗				✗		946 (19%)
13		✗	✗	✗	✗					616 (12%)
14		✗	✗	✗	✗	✗				836 (17%)
15		✗	✗	✗	✗	✗	✗			859 (17%)
16		✗	✗	✗	✗	✗	✗	✗		1126 (23%)
17		✗	✗	✗	✗	✗	✗	✗	✗	1132 (23%)
18	Original Version									946 (19%)

Table 3. Results obtained for the correct assignment of the pairs (lemma + tag) to the unknown words by changing the modules used in the processing of the excerpt with 4000 sentences.

In Table 3 is presented the results of processing the excerpt with 4000 sentences. For each column in Table 3 correspond the module used in each test realized: Off - all modules off, except *Termination* that is always active; DS - Direct Substitution; C - Compounds; UC - Upper Case Converter; H - Hyphen; LM - Language Model; P - Phonemes; D - Diacritics; S - Select Alternative.

It is conclusive that the modules that contribute with more identified words is *Language Model* and *Diacritics*.

As presented, it is conclusive that the modules that contribute with more identified words is *Language Model* and *Diacritics*. The new solution for *Guesser* will produce significantly better results when comparing with the original. The new solution can correct identifying more 4% of unknown words.

5 Conclusions

This work addresses the problem of improve *Guesser* module in the identification of words that are not in *LexMan* lexicon. To achieve this was added different strategies to the original solution to create more alternatives for the unknown words. The results obtained in comparing with the baseline was better regarding the precision in unknown words identification than was for processing time, nevertheless the time was better when processing big texts.

In general, this work improved the computation performance of the *Guesser* module.

References

1. Almeida, H., Baptista, J., Mamede, N.: Suffix identification in Portuguese using transducers. In: Proceedings of INFORUM 2016 – Simpósio de Informática. pp. 38—47. INForum 2015, IST, Lisboa, Portugal (2015)
2. Bick, E.: The Parsing System "PALAVRAS". Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. Phd thesis/tese de doutoramento, University oh Aarhus, Aarhus, DF (December 2000)
3. Diniz, C., Mamede, N.: Lexman - lexical morphological analyser. Tech. rep., L2F / INESC ID Lisboa, Lisboa (2011)
4. Diniz, C., Mamede, N.J., Pereira, J.C.S.D.: RuDriCo2 - A faster disambiguator and segmentation modifier. In: II Simpósio de Informática (INForum). pp. 573–584. Universidade do Minho, Portugal (September 2010)
5. Hagège, C., Tannier, X.: XRCE-T: XIP temporal module for TempEval campaign. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 492–495. SemEval '07, Association for Computational Linguistics, Stroudsburg, PA, USA (2007)
6. Karin Beijering, C.G., Heeringa, W.: Predicting intelligibility and perceived linguistic distance by means of the levenshtein algorithm pp. 4–6 (2008)
7. Mamede, N., Baptista, J., Cláudio, D.: STRING - An Hybrid Statistical and Rule-Based Natural Language Processing Chain for Portuguese. In: Springer (ed.) PROPOR 2012 (2012)
8. Medeiros, J.C.D.: Processamento Morfológico e Correcção Ortográfica do Português. Dissertação para obter o grau de mestre em engenharia electrotécnica e de computadores, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa, IST (Fevereiro 1995)
9. Parlamento: Debates Parlamentares - 3ª República (1976-2015), http://debates.parlamento.pt/catalogo/r3/dar
10. Ribeiro, R.: Anotação Morfossintáctica Desambiguada do Português. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa (Março 2003)
11. Rocha, P., Santos, D.: Cetempúblico: Um corpus de grandes dimensões de linguagem jornalística portuguesa. In: Actas do V Encontro para o processamento computacional da língua portuguesa escrita e falada, PROPOR'2000. pp. 131–140. Atibaia, são Paulo, Brasil (November 2000)
12. Sumit Sharma, S.G.: Procedia Computer Science. In: A correction model for real-word errors. pp. 99–106. Elsevier B. V., Kurukshetra, Haryana, India (December 2015)

13. Vicente, A.M.F.: LexMan: um Segmentador e Analisador Morfológico com Transdutores. Master's thesis, Instituto Superior Técnico (June 2013)
14. Viterbi, A.J.: Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm. Institute of Electrical and Electronic Engineers (IEEE) Transactions on Information Theory **13**(2), 260–269 (1967)