Very high accuracy rule-based nominal lemmatization with a minimal lexicon

António Branco and João Silva

University of Lisbon, Department of Informatics
NLX–Natural Language and Speech Group
http://nlx.di.fc.ul.pt

1 Introduction

In natural language processing, lemmatization is a procedure by which an inflectionally normalized form (the lemma) is automatically assigned to word forms. The particular task that is addressed in this paper is that of Nominal Lemmatization, targeting only Adjectives and Common Nouns.

Note that the other tokens from the other nominal categories — such as Articles, Demonstratives, etc. — form a closed list, and can thus be easily lemmatized by a simple list look-up procedure. In this paper, our concern will thus be the tokens from the open nominal categories.

This paper describes a shallow processing, rule-based algorithm for Nominal Lemmatization in Portuguese with minimal word lists. Additionally, evaluation results are presented scored from an efficient implementation of this algorithm.

In Section 2, we describe the lemmatization task in greater detail and the issues that it raises. In Section 3, we outline the shallow processing algorithm that is used while Section 4 deals with the methods used to minimize the lexicon that is required. In Section 5, we present some of the harder cases which are caused by ambiguity. In Section 6, evaluation results, together with details about the performance of the implementation, are presented. In Section 7 we provide links to on-line demos and services that use the lemmatizer. Finally, in Section 8, the results are discussed and some prospects for future work are presented.

2 Description of the problem

By providing a normalized representation for word forms, the lemma is an extremely important piece of linguistic information for the subsequent processing stages, in particular for those involving syntactic and semantic processing. The main motivation for doing lemmatization is thus to allow for more efficient searches in a lexicon: instead of having to list every inflected form of a word, the lexicon only has to include an entry for the lemma.

In this regard, lemmatization is similar to the task of stemming, where several different word forms are also conflated into a single form. This seems to lead to some confusion in terminology, and sometimes a task is considered to be lemmatization when, in fact, it is stemming. In (Silva & Oliveira, 2001), for instance, the authors claim to study several "lematizadores" for Portuguese which are actually stemmers.

However, stemming and lemmatization are quite different tasks. Their major difference, which must be underlined, is this: in stemming, it is not necessary for the resulting form (the stem) to be a genuine word, as stemming is typically used for Information Retrieval (IR), and its purpose is only to group "similar" words. As Porter (1980) states, "[T]he suffixes are being removed simply to improve IR performance, and not as a linguistic exercise."

The lemma, however, is not simply an ad-hoc sub-string that is common to various word forms, but a word form in a conventionalized "format." In Portuguese, the lemma for Adjectives and Common Nouns is the masculine singular form, if it exists. If this form does not exist, the lemma is the singular form. If this form also does not exist, the lemma matches the word form. Some cases illustrating these conventions are shown below:

`gatas`	(fem. plu.)	→	`gato`	(*Eng.: cat*, masc. sing.)
`cobras`	(fem. plu.)	→	`cobra`	(*Eng.: snake*, fem. sing.)
`termas`	(fem. plu.)	→	`termas`	(*Eng.: spa*, fem. plu.)

The lemma corresponds thus to the form that is found in a dictionary for a given word and the key problem one has to tackle in a computational approach to lemmatization is the inevitable incompleteness of a lexicon, regardless of that lexicon containing form-lemma or lemma-form mappings.

A form-lemma lexicon involves an exhaustive listing of every possible word form and its corresponding lemma. A lemma-form lexicon — the approach followed in Jspell (Almeida & Pinto, 1994; Simões & Almeida, 2000) — captures inflection regularities through rules. It uses a dictionary of lemmas, where each lemma is associated to a transformation rule that allows the generation of new word forms. To find the lemma for a given word form, one looks for a lemma in the dictionary which, through its respective rule, can originate the given word form. Neither approach is practically viable since the lexicon, being open to the inclusion of new words, cannot be definitely complete.

Thus, when addressing the task of nominal lemmatization, it is worth noting that the direct approach of assigning lemmas by means of a mere lexical look-up is far from being the most convenient methodology. We need an approach that is able to seamlessly handle unknown or new words.

3 Algorithm outline

To implement the nominal lemmatizer, we build upon the morphological regularities found in word inflection and use a set of transformation rules that "undo" the form changes due to inflection.

As it can be seen in Figure 1, a single transformation rule can encapsulate a large number of lemma assignments, which would otherwise have to be explicitly listed.

Rule	Covered cases
-ta → -to	...
	aberta → aberto
	adulta → adulto
	alta → alto
	...

Figure 1: A transformation rule

A transformation rule will thus cover most cases for the corresponding termination. There will be, however, exceptions that should be accounted for. For example, the word porta (*Eng.: door*) is a feminine Common Noun whose lemma is porta but applying the rule from Figure 1 would give the lemma porto. Exceptions such as this have to be collected for each rule.

Thus, the basic rationale for the rule-based lemmatizer described here is to gather a set of transformation rules that, depending on the termination of a word, replace that termination by another, and complement this set of rules with a list of exceptions. Neologisms are expected to comply with the regular morphology and are accounted for by the rules.

3.1 Transformation rules

Transformation rules are replacement rules used to "undo" the morphographemic changes caused by the inflection process. So, for instance, if words ending in -to are typically inflected into words ending in -ta to obtain the feminine form, the reverse transformation should be present in the lemmatization rules.

For example, doing this for all four possible Gender and Number combinations one can obtain for the words ending in -to (viz. -to, -tos, -ta and -tas) leads to the set of rules exemplified in Figure 2.

-to (masc. sing.)[1]
-tos (masc. plu.) → -to
-ta (fem. sing.) → -to
-tas (fem. plu.) → -to

Figure 2: "Direct" lemmatization rules

These can be easily implemented by creating a procedure that, for each given word, scans the set of rules for a match, i.e. a rule whose left-hand side matches the termination of the given word. Upon finding a rule, the corresponding termination of the word is replaced by the string on the right-hand side of the rule.

[1] Words ending in -to do not need to be transformed, since they are already in the desired form.

3.2 Exceptions

To collect exceptions we resort to machine-readable dictionaries (MRD) that allow searching for words on the basis of their termination. Given that dictionaries include lemmas, and never regular inflected forms, it is possible to collect the needed exceptions simply by searching for words with terminations matching the termination of inflected words.

For instance, to find exception to the rule `-ta → -to`, we search for words ending in `-ta`. Since the dictionary does not list inflected forms, every word ending in `-ta` that is found must be an exception.

These collected exceptions are entered into an internal word list of the lemmatizer and each is coupled with its specific transformation rule.

Exceptions whose lemma matches the word form are associated to a "dummy" transformation that does not change the word form. For example, `porta` is associated with a `-ta → -ta` transformation.

The size of the list of exceptions may reach several hundreds of entries. Although this might seem at first blush a large size, one should bear in mind the following points:

- The number of words covered by a transformation rule is much larger than the number of exceptions one needs to collect for that rule.
- New words that enter the lexicon tend to follow the general rule for inflection and, consequently, for lemmatization. As a result, the list of exceptions is considerably stable and needs to be less frequently updated than an exhaustive look-up table of word forms and their lemmas.

Note also that while we use the term "list of exceptions," these are not stored internally in a linear list, since this is a notoriously inefficient data-structure. The lemmatizer uses a balanced binary tree to store exceptions, allowing for an efficient look-up of a large number of entries.

4 Minimizing the lexicon

Using the algorithm outlined in the previous chapter, one drastically reduces the size of the lexicon that is necessary, since only exceptions have to be explicitly listed. Nevertheless, one of our objectives is to reduce the lexicon as much as possible. The techniques we use for doing this are addressed in the next sections.

4.1 Rule hierarchy

The number of exceptions one needs to collect for a given rule can be decreased as extra, more specific rules can be used to capture regularities found within the exceptions themselves.

For instance, terminations matching `-ia` are usually transformed into `-io` (feminine into masculine) to obtain the lemma. When collecting the exceptions for this rule one can find, among others, the whole family of words ending in the Greek suffix

-fobia (*Eng.: -phobia*) whose lemma matches the word form. So, instead of listing them as exceptions for the -ia → -io rule, one can simply add the new rule -fobia → -fobia to handle all those cases.

In terms of the implementation of the algorithm, note that to allow for this, the choice of which transformation rule to apply must use the longest match criteria, so that the longer, more specific termination will necessarily be chosen.

4.2 Recursive, single-step rules

The set of lemmatization rules shown in Figure 2 transforms any of the various inflected word forms from the -to "family" directly into the corresponding lemma.

By doing this transformation directly, special care must be taken when collecting the exceptions for the -tas → -to rule since a dictionary search for words ending in -tas will not provide all the necessary exceptions.

This happens because these exceptions must also include the words that are exceptions to the -ta → -to rule but when inflected for plural.

For example, the word porta, as seen before, is one of the exceptions to the -ta → -to rule. Consequently, its plural form, portas (*Eng.: doors*), is an exception to the -tas → -to rule, but — being an inflected form — it will not be listed in a dictionary.

Obtaining the exceptions in plural form is rather easy as one needs only apply a simple transformation to every exception in singular form (e.g. -ta → -tas). There is, however, an easier way of handling these cases, namely through the use of recursive, single-step lemmatization rules, which is an approach that does not require extending the exceptions list.

Single-step rules are transformations that only affect a single feature (Gender or Number). From the set of rules shown above, the last one (viz. -tas → -to) is not a single step-rule, since it transforms a feminine plural form into a masculine singular form in a single transformation step.

The set of direct rules from Figure 2 can be rewritten into the set of single-step rules seen below, in Figure 3. The single-step rules are similar to the set of direct rules above, differing only in the rule for feminine plural word forms, which now transforms these words into their feminine singular form.

-to (masc. sing.)
-tos (masc. plu.) → -to
-ta (fem. sing.) → -to
-tas (fem. plu.) → -ta

Figure 3: Single-step lemmatization rules

However, when the word that is to be lemmatized is in the feminine plural form, a single-step transformation might not be enough to produce a lemma. Recursive lemmatization rules are repeatedly applied, transforming a word into another, until an exception has been found or until there is no rule that can be applied.

```
adultas → adulta → adulto
portas → porta
```

Figure 4: Recursive application of single-step rules

For example, `adultas` (*Eng.: adults*, feminine plural) would firstly be transformed into `adulta` (*Eng.: adult*, feminine singular) — and, in a second transformation step, into `adulto` (*Eng.: adult*, masculine singular), at which point no more rules are applicable and that form is returned as being the lemma — while `portas` would be transformed into `porta`, matching the exception and returning that form as the lemma.

By using recursive single-step rules, it is not necessary to extend the exceptions list with inflected forms of exceptions. In addition, it is straightforward to extend the algorithm to apply such rules: it is sufficient to run the transformation procedure on its own output until no rule can be applied or until an exception is found.

4.3 Handling non-inflectional affixes

Besides its various inflected forms, a word can also have non-inflectional affixes, greatly increasing the number of possible forms that the lemmatizer must handle.

In this section, we discuss how to handle non-inflectional prefixes and suffixes and how to resolve the complexity that arises when both occur simultaneously in the same word form.

4.3.1 Stripping prefixes

Prefixed words raise a difficulty when dealing with the search for exceptions to a transformation rule.

Taking again `porta` as an example of an exception, every word formed by combining it with a prefix is also an exception to the `-ta → -to` rule, like `anteporta`, `autoporta`, etc. This entails that, in addition to `porta`, all words formed by prefixing `porta` must also be included in the exceptions to the `-ta → -to` rule.

Adding all such prefixed words to the list of exceptions is not an appropriate approach, as the number of prefixes, even though not unlimited, is still very large and, more important, prefixes can be accumulate, as in `autosuperporta`.[2]

As a better solution to this problem, the algorithm is designed to temporarily remove prefixes from words when obtaining the lemma. After getting the lemma, the prefixes that were removed are returned back to their original place.

This requires a list of possible prefixes. When the starting characters of a word match one of the listed prefixes, these characters are temporarily removed. The process is repeated until it is not possible to remove any further prefixes.

It is important to note that, before removing any prefix, the word form is checked against the exceptions list. This is done to account for word such as `antena` (*Eng.:*

[2] Even though some of these prefixed forms might be unusual, they are perfectly valid from a purely morphological point of view.

antenna) which, although beginning with a sequence of characters matching a listed prefix (cf. `ante-`), it is not a prefixed word.

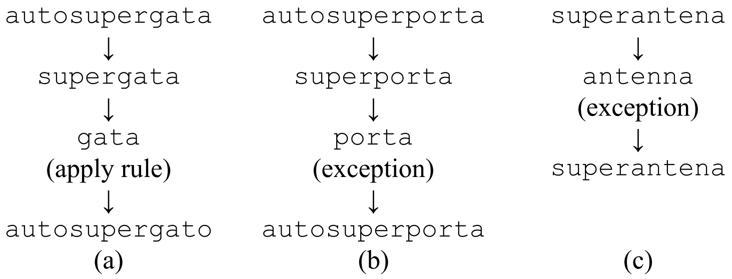

Figure 5: Prefix stripping

The prefix stripping process is illustrated in Figure 5 by means of three examples:

(a) `gata` (*Eng.: cat*, feminine singular), a common noun which should be lemmatized into the masculine singular form `gato`. After stripping `auto-` and `super-`, one gets `gata`, which has no more prefixes to get stripped. The `-ta` → `-to` rule can then be applied, giving the result `gato`.

(b) `porta`, an exception, which should be lemmatized into `porta`. After stripping both prefixes, one gets `porta`, which is one of the listed exceptions.

(c) `antena`, an exception, which should be lemmatized into `antena`. This is similar to the previous case, but it illustrates a case where a possible prefix (`ante-`) was not removed, as the word form was listed in the exceptions list.

4.3.2 Degree suffixes

Degree suffixes (for Diminutive and Superlative) seem, at first, to be easily handled by the same mechanism used for inflection suffixes but specific difficulties should be taken into account.

For instance, `gatinho` (*Eng.: [small] cat*) is easily lemmatized into `gato` by a `-inho` → `-o` rule. Also, any exceptions to this rule, like `vizinho` (*Eng.: neighbor*) are easily collected by using MRDs. In addition, as seen before, to minimize the number of exceptions one needs to collect, single-step recursive rules can be used, such as `-inha` → `-inho`, to account for inflected forms, like `gatinha` (*Eng.: [feminine small] cat*).

However, there are some situations that cannot be as easily handled by this mechanism. For the sake of concreteness, we will take the words `vizinha` (*Eng.: [feminine] neighbor*) and `portinha` (*Eng.: [small] door*) as an example of such a situation.

- Having the `-inha` → `-inho` rule described above avoids having to list `vizinha` explicitly as an exception. But, by using this rule, the word `portinha` would be transformed into `portinho` and, in a second step, lemmatized as porto. To prevent this from happening, one has to list portinha explicitly as an exception to the -inha → -inho rule.

- As an alternative, one could instead use -inha → -a as a default rule, which would correctly lemmatize portinha. However, in this case, we would need to list vizinha as an exception, to prevent it from being lemmatized into viza.

Accordingly, regardless of which rule is chosen to be the default one, one has to list extra exceptions. To avoid this, a possible approach is to allow for a rule to branch out into several possible transformations, thus giving rise to a branching search for the solution.

4.4 Branching search

To allow branching search of lemmas, the rules are extended to allow for various alternative transformations. Each alternative opens up a new search branch. The lemmatization proceeds in parallel through each branch until no further rules are applicable or until an exception has been found.

The heuristic for the choice of lemma takes the leaf node found at the lowest search depth (i.e. the result that required the fewest transformation steps).

For instance, to handle the problem described above, found when dealing with portinha and vizinha, we create a -inha → -a,-inho rule is created. For words ending in -inha, this rule creates two search branches. For the first branch, the -inha → -a is applied, while the search in the second branch proceeds under the -inha → -inho transformation. The result found at the lowest search depth (vizinho for vizinha and porta for portinha) is taken as being the lemma. Both these search trees can be seen in Figure 6.

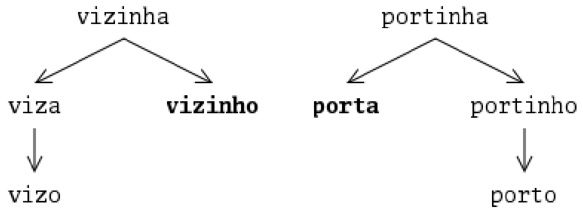

Figure 6: Branching search

The branching search has the added advantage of seamlessly handling words that combine multiple prefixes and suffixes.

When several non-inflectional affixes occur in a single word, one must follow all possible paths of affix removal/transformation to ensure that an exception is not overlooked.

As an example, take the lemmatization of anteninha (*Eng.: [small] antenna*), illustrated in Figure 7.

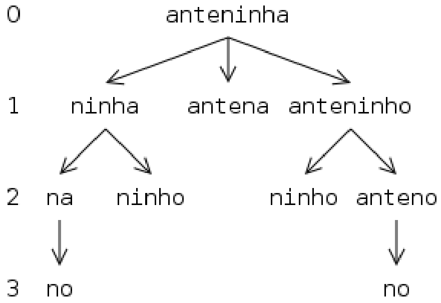

Figure 7: Lemmatizing anteninha

Both ante- and -inha are possible affixes. In the first step, three search branches are opened. The first branch (ninha) corresponds to the removal of the ante- prefix, while the two other branches (antena and anteninho) are the result of applying the transformations in the -inha → -a,-inho rule. The lemma is the leaf node with the lowest search depth (i.e. obtained with the fewest steps), antena.

The search under several branches seems to lead to a great performance penalty, but one must bear in mind that only a few words have non-inflectional affixes, and most of those have only one, in which case there is no branching at all. So, in practice, the branching search does not incur in a damaging performance penalty while allowing for arbitrarily complex affixed words.

5 Ambiguity

At this point, it is also worth noting that there are some cases were the lemma does not depend solely on the word form.

5.1 Sense dependency

Lemmatization may be conditioned by the part-of-speech (POS) of the word form. For example, ética, when occurring as an Adjective, is the feminine singular form of *ethical* and should therefore be lemmatized into ético, the masculine singular form. However, when occurring as a Common Noun, ética has the meaning of *ethics* and must be taken as an exception to the -ca → -co rule.

To cope with this, these items are specifically listed and the lemmatizer is assumed to run over a POS-tagged text.

More problematic is the case where lemmatization is dependent on the sense of the token at stake. For instance, the word copas may refer to the *hearts* suit of playing cards, in which case its lemma is copas, or it may be the plural form of copa (*Eng.: cupboard*, *treetop*), in which case it should be lemmatized into the singular form copa. Ambiguous words such as this cannot be resolved by any lemmatizer to be applied before word sense disambiguation (WSD).

5.2 Words ending in -e

Words ending in `-e` also pose a problem since these words do not have a preferential Gender feature (Mateus *et al.*, 2003, pp. 923). Consequently, when such words receive a non-inflectional suffix, their termination matches that of words ending in `-a` or `-o`. For instance, the diminutive form of the feminine singular word `parede` (*Eng.: wall*) is `paredinha`. When lemmatizing this form, there are two plausible transformations, viz. `-inha → -a` and `-inha → -e`.

These problematic words are handled as exceptions, i.e. `paredinha` is listed as an exception to the `-inha → -a` rule.

5.3 Hyphenated compound words

Hyphenated compound words are formed by morphological or by morphosyntactic composition (Mateus *et al.*, 2003, pp. 971). The latter type, in particular, cannot be lemmatized by simply replacing the termination of the compound word (i.e. replacing the ending of the string). For instance, take the following examples:

```
aluna-modelo → aluno-modelo
patos-bravos → pato-bravo
surdas-mudas → surdo-mudo
```

These examples seem to suggest that a straightforward way of lemmatizing such words is to lemmatize each part of the compound separately (e.g. the lemma of `surdas` is `surdo` and the lemma of `mudas` is `mudo`). While this approach works for the examples above, there are cases where it does not. For instance:

```
abre-latas   → abre-latas
arranha-céus → arranha-céus
guarda-redes → guarda-redes
```

These particular cases behave differently because they are formed through a different composition process ("estruturas de reanálise," (Mateus *et al.*, 2003, pp. 982)), where the first element of the compound is a verb form. The difficulty in handling these cases stems from the fact that the shallow processing lemmatization process has no automatic way of determining the type of process that originated a given hyphenated compound word (i.e. it is not possible to determine if `guarda` in `guarda-redes` is a verb form or a noun). Consequently, at present, the lemmatizer can only handle these cases through a plain lexical look-up.

5.4 Alterations in diacritics

When a word receives a non-inflectional suffix, diacritic marks possibly present in the word may be removed or change position. For instance, the superlative of `rápido` (*Eng.: quick*) is `rapidíssimo` (the `á` loses its diacritic mark). A rule such as

-íssimo → -o can “undo” the superlative, but it does not restore the diacritic. The ambiguity stems from the fact that there is no morphological reason to disregard the word rapido (without the diacritic mark) as a possible lemma.

Currently, these cases are considered as exceptions, e.g. rapidíssimo is listed as an exception to the -íssimo → -o rule and assigned rápido as the lemma.

6 Evaluation

In order to implement the algorithm, a list of 126 transformation rules was necessary. The list of exceptions to these rules amounts to 9,614 entries. Prefix removal is done resorting to a list of 130 prefixes.

The lemmatizer was evaluated over the 50,637 Adjectives and Common Nouns present in a 260,000 token corpus consisting of manually annotated newspaper excerpts and short novels.[3] Given that the POS tags are already assigned, the lemmatizer does not need any window of context around the target token to be evaluated. Hence, evaluating it over a list of the target tokens is the same as evaluating over the corpus.

In this evaluation list there are 203 tokens that are semantically ambiguous. As discussed above, the lemmatizer is not able to handle these cases, since for these tokens to be lemmatizer a WSD step is necessary. Therefore, this figure helps to suggest an upper-bound of 99.60% to the recall of a lemmatizer based on shallow processing.

On the remaining 50,434 tokens, there were 1,072 lemmatization errors, yielding a precision of 97.87%. It is also useful to look at the F-measure, which combines precision and recall into a single value (Manning & Schütze, 1999, pp. 269). We give equal weights to precision (p) and recall (r), in which case the formula for calculating the F-measure can be simplified to $f = 2pr/(p+r)$. These results are summarized in Table 1.

recall	***precision***	***f-measure***
99.60%	97.87%	98.73%

Table 1: Lemmatizer evaluation

Most of these errors are caused by words that are missing from the exceptions list and, therefore, receive the wrong lemma.

To the best of our knowledge, for the few tools for the processing of Portuguese that are claimed to involve some step of nominal lemmatization, e.g. (Almeida & Pinto, 1994), (Simões & Almeida, 2000) or (Bick, 2000), no evaluation results for that step in isolation are provided. No direct comparison with other eventual nominal lemmatizers of Portuguese seems thus to be possible at present. In this respect, and to gain a sense of the level of performance of the lemmatizer described here, one can resort to recent, top accuracy lemmatizers for other languages with similar nominal inflection systems, such as Spanish. That is the case of the (nominal and verbal) lemmatizer presented in (Chrupala, 2006), which reportedly outperforms other lemmatizers for Spanish, with an

[3] We are grateful to CLUL (Centro de Linguística da Universidade de Lisboa) for their help with this corpus.

F-measure of 92.48%. Tough this is a tool for a language other than Portuguese, and handles both verbal and nominal lemmatization, it is the best hint at hand to get the sense that the nominal lemmatizer for Portuguese described here is performing at state of the art level of accuracy, and it is most likely the best such tool described in the literature so far.

As for speed performance, the lemmatizer takes 0.32 seconds to complete its task when running over the evaluation data of 10,581 items (corresponding to ca. 33,000 word/sec on average). The computer running the lemmatizer has an Intel Pentium IV processor with a 3.0 GHz clock speed.

7 On-line services

The lemmatizer is part of the LX-Suite (a set of NLP tools, including a tokenizer, a POS tagger and a verbal lemmatizer), which can be seen in an on-line demo at `http://lxsuite.di.fc.ul.pt`. The lemmatizer also supports an on-line service for the inflection of nominal forms (Common Nouns and Adjectives), which can be found at `http://lxinflector.di.fc.ul.pt`.

8 Concluding remarks

This paper presented a shallow processing, rule-based algorithm for the lemmatization of Adjectives and Common Nouns in Portuguese.

The algorithm builds upon the morphological regularities found in the inflection process by using a set of transformation rules that replace the endings of words. These rules are supplemented by a list of exceptions. The algorithm also uses a variety of methods to reduce the number of exceptions one has to store to a minimum: (i) hierarchical rules, (ii) single-step recursive rules, (iii) affix stripping and (iv) branching search.

The lemmatizer is not able to handle words that are semantically ambiguous. Such cases, however, have proved to be rare, and the lemmatizer still achieves a very high score for recall. Future work will focus on improving the precision score also by expanding the exceptions list with new entries that may have escaped until now.

References

Almeida, José & Ulisses Pinto (1994). Jspell – Um módulo para análise léxica genérica de linguagem natural. In *Proceedings of the 10th Encontro Anual da Associação Portuguesa de Linguística (APL).*

Bick, Eckhard (2000). The parsing system PALAVRAS: Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. PhD thesis, University of Århus, Denmark.

Chrupala, Grzegorz (2006). Simple data-driven context-sensitive lemmatization. In *Proceedings of Sociedad Española para el procesamiento del Lenguaje Natural*

(SEPLN).

Manning, Christopher & Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mateus, Maria Helena Mira, Ana Maria Brito, Inês Duarte, Isabel Hub Faria, Sónia Frota, Gabriela Matos, Fátima Oliveira, Marina Vigário & Alina Villalva (2003). *Gramática da Língua Portuguesa*. Editorial Caminho.

Porter, Martin (1980). An algorithm for suffix stripping. *Program* 13 (3), pp. 130—137.

Silva, Gilberto & Claudia Oliveira (2001). *Lematizadores com base em léxico*. Technical report 069/DE9/01. Departmento de Engenharia de Sistemas do Instituto Militar de Engenharia.

Simões, Alberto & José Almeida (2000). jspell.pl – Um módulo de análise morfológica para uso em processamento de linguagem natural. In *Proceedings of the 16th Encontro Anual da Associação Portuguesa de Linguística (APL)*, pp. 485—495.