Assessing the Efficiency of Suffix Stripping Approaches for Portuguese Stemming

Wadson Gomes Ferreira, Willian Antônio dos Santos, Breno Macena Pereira de Souza, Tiago Matta Machado Zaidan, and Wladmir Cardoso Brandão

Department of Computer Science,
Pontifical Catholic University of Minas Gerais, Belo Horizonte, Brazil
{wadson.gomes,willian.santos.838460,breno.macena,tiago.zaidan}@sga.pucminas.br
{wladmir}@pucminas.br

Abstract. Stemming is the process of reducing inflected words to their root form, the stem. Search engines use stemming algorithms to conflate words in the same stem, reducing index size and improving recall. Suffix stripping is a strategy used by stemming algorithms to reduce words to stems by processing suffix rules suitable to address the constraints of each language. For Portuguese stemming, the RSLP was the first suffix stripping algorithm proposed in literature, and it is still widely used in commercial and open source search engines. Typically, the RSLP algorithm uses a list-based approach to process rules for suffix stripping. In this article, we introduce two suffix stripping approaches for Portuguese stemming. Particularly, we propose the hash-based and the automata-based approach, and we assess their efficiency by contrasting them with the state-of-the-art list-based approach. Complexity analysis shows that the automata-based approach is more efficient in time. In addition, experiments on two datasets attest the efficiency of our approaches. In particular, the hash-based and the automata-based approaches outperform the list-based approach, with reduction of up to 65.28% and 86.48% in stemming time, respectively.

1 Introduction

The Web comprises over 30 trillion uniquely addressable documents. Recently, the leading commercial search engine reported the processing of more than 3.3 billion user queries each day [4]. The massive-scale nature of the Web demands efficient tools to manage, retrieve and filter information [2]. In this environment, time and retrieval performance are critical for search engines to provide an effective search experience to their users.

Typically, search engines use stemming to reduce inflected words to their stem, i.e., the portion of the original word that remains after affixes removal [2]. For instance, the word "connect" is the stem of the words "connected", "connecting", "connection" and "connections". Stemming document corpus reduces the vocabulary size and hence, the time to process queries. In addition, query reformulation by stemming increases the match between the query and relevant documents in the corpus, ultimately improving the recall. Due to the increasing size of the Web and the increasing query traffic on search engines, a particularly challenging information retrieval problem is the development of effective stemming approaches. For Portuguese stemming, the RSLP [7] is an effective algorithm widely used in commercial and open source search engines. For instance, Lucene[1] uses the RSLP list-based approach for suffix stripping.

In this article, we propose two suffix stripping approaches for Portuguese stemming, the hash-based and the automata-based approaches. Additionally, we assess their efficiency by contrasting them with the state-of-the-art list-based approach [7]. In particular, the list-based approach compares word suffixes to a well defined list of suffix rules, iteratively reducing the word. At the end, the reduction performed by the last rule in the list produces the stem. However, usually only a few suffix rules need to be processed to reduce a word. From this observation, we propose the hash-based approach, which breaks the single big list of suffix rules in small ones, deciding which list should be processed based on the word suffixes, thus reducing the number of suffix rules to check. In addition, we also propose the automata-based approach which uses a deterministic finite automata (DFA) to iteratively check word suffixes to decide when to perform a reduction or not, avoiding unnecessary suffix rules checking.

Complexity analysis shows that the time complexity is constant for the automata-based approach, sub-linear for the hash-based approach, and linear for the list-based approach, considering the number of suffix rules to process. Additionally, the space complexity is constant for the automata-based approach and linear for the hash-based and list-based approaches. However, in addition to store the list of suffix rules, the hash-based approach stores a hash table to distribute the suffix rules of the list by word suffixes, thus consuming more space than the list-based approach.

Experiments using two datasets attest the efficiency of our approaches. The hash-based and the automata-based approaches outperform the list-based approach baseline with reduction of up to 65.28% and 86.48% in stemming time, respectively. The key contributions of this article can be summarised as follows:

- We propose two suffix stripping approaches for Portuguese stemming. The proposed approaches are simple and can be adapted to handle other languages, such as English and Spanish.
- We provide complexity analysis for the list-based approach baseline and our approaches, highlighting the differences in space and time complexity between them;
- We thoroughly evaluate our approaches contrasting them with the state-of-the-art baseline from literature [7].

The remainder of this article is organised as follows: Section 2 reviews the related literature on stemming algorithms. Section 3 describes our suffix stripping approaches for Portuguese stemming, as well as the baseline, presenting

[1] `http://lucene.apache.org`.

their complexity analysis. Section 4 describes the setup and results of the experimental evaluation of our approaches. Finally, Section 5 provides a summary of the contributions and the conclusions made throughout the other sections, presenting directions for future research.

2 Related Work

Stemming algorithms use four strategies to produce the stem for a word: table lookup, successor variety, n-grams, and affix removal [2]. The table lookup strategy performs a simple lookup of the word in a table to retrieve its stem. Despite simple, this strategy is strongly dependent on the language vocabulary. The sucessor variety is a complex strategy, which identifies morpheme boundaries within the language organization to produce the stem. The n-grams strategy is more a word clustering procedure than a stemming strategy, since it simply identifies bigrams and trigrams in the text. The affix removal strategy removes or replaces affixes of the word, following well defined affix rules for the language. In particular, the affix removal strategy is simple, intuitive, effective, and less dependent on the language vocabulary. Mostly, it includes suffix stripping.

For Portuguese, there are three suffix stripping algorithms reported in literature, an adapted version of the Porter stemmer for the English language [8], and two others specifically designed for the Portuguese language: the RSLP [7], and the STEMBR [1]. The Portuguese Porter stemmer is a translation of the English version for Portuguese, resulting in the absence of key reductions, such as feminine and degree. The STEMBR algorithm processes a set of rules for suffix removal in a specific order, defined from statistical analysis on a corpus of documents, with a balanced word distribution for a language. The performance of the algorithm depends on the quality of the corpus.

The RSLP was the first suffix stripping algorithm for Portuguese stemming. It processes a set of well defined suffix rules using an exception list to address language irregularities, ultimately providing an effective word reduction [6]. Different from STEMBR, the RSLP algorithm does not need to define an order to process the suffix rules from a corpus of documents. In addition, its set of suffix rules includes feminine and degree reduction. Thus, given the effectiveness and simplicity of the RSLP algorithm, the suffix stripping approaches we investigate in this article are based on it. The complexity analysis of the RSLP algorithm [3] shows that the time complexity, in the worst case, can be expressed by

$$n * W * \sum_{i=1}^{R} (S_i * E_i) \tag{1}$$

where n is the number of processed words, W is the size, in characters, of the longest word, R is the number of suffix rules, S_i is the size, in characters, of the suffix rule i, and E_i is the size of the exceptions list of i. Considering that lookups in the exceptions list are performed in constant time, the time complexity to reduce a word j can be expressed by:

$$W_j * \sum_{i=1}^{R} S_i \tag{2}$$

From the Equation 2, we observe that the number of suffix rules that should be compared to the word suffix to perform reductions, in the worst case, is R, i.e., all the suffix rules should be compared to conclude that no reduction should be performed. In this case, the complexity is $O(W_j R)$, or simply $O(W_j)$, if we consider R constant. The point is that the value of R significantly impacts in time, since the greater the R, the greater the number of comparisons between characters in the suffix rules and word suffix must be performed. From this observation, in the next section we present three different approaches to perform suffix stripping, one corresponding to the list-based approach [3], and the other two corresponding to our proposed approaches, which improve the efficiency by reducing the amount of processed rules, i.e., the value of R.

3 Suffix Stripping Approaches

The RSLP algorithm is composed by eight steps, of which six are word reductions based on suffix stripping, i.e., removing or replacing a piece of the word, the suffix. Figure 1 presents the steps of the RSLP stemming algorithm.

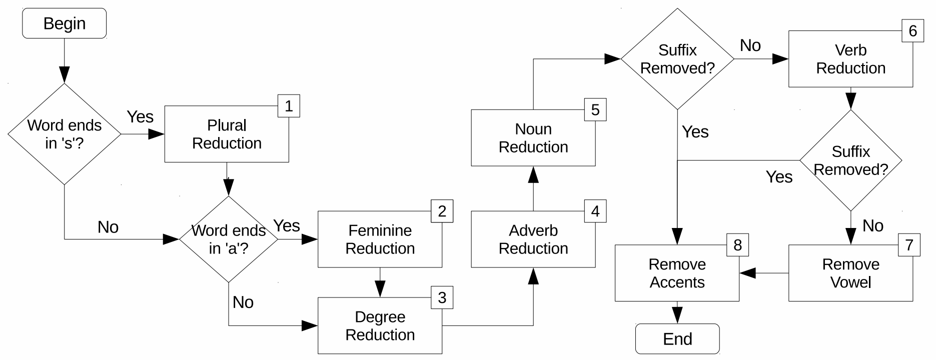

Fig. 1. The steps of the RSLP stemming algorithm for Portuguese

From Figure 1, we observe that suffix stripping are performed from step 1 to 6. The challenge in suffix stripping is not only to decide performing word reductions, but mostly avoid unnecessary decisions. In particular, reducing decisions are made by comparing word suffixes to suffix rules, where each rule is represented as a tuple composed by: i) the suffix to be replaced or removed from the word;

ii) the minimum size of the word to perform the reduction; iii) the new suffix in case of replacement, or empty in case of removal, and; iv) the exceptions list, i.e., the list of words that should supposed be reduced by the rule, but should not be reduced.

Sometimes, there are more than one suffix rule for the same reduction. This occurs when one suffix is contained in another, e.g., for plural reduction in the word “mares”, the suffix “s” is contained in the suffix “es”. In this case, the suffix stripping approach should perform only the reduction related to the longest suffix [7]. Considering the aforementioned example, only the “es” suffix should be removed from the word, reducing it from “mares” to “mar”.

3.1 The List-Based Approach

The list-based approach provides a single list of suffix rules to iteratively compare suffixes, one by one, to the word suffix and decides reducing the word or not. Figure 2 presents an example of a list of suffix rules for Portuguese. The complete list of suffix rules is presented by Orengo and Huyck [7].

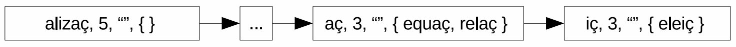

Fig. 2. Example of a list of suffix rules for Portuguese

For each suffix rule in the list, the list-based approach first checks if the suffix of the rule matches the word suffix. Second, it checks if the word is not in the exceptions list of the suffix rule. Third, it checks if the size of the resulting word is larger than or equal the minimum word size stated in suffix rule. Finally, it performs the word reduction, removing the word suffix or replacing it by the new suffix in the rule.

3.2 The Hash-Based Approach

The hash-based approach provides a hash table, where each hash entry is composed by a character key, and a pointer to a list of suffix rules. Figure 3 presents an example of a hash table with hash entries pointing to lists of suffix rules.

From the word to be reduced, the hash-based approach takes the last character to lookup the hash table and, in case of a negative lookup no reduction is performed. In case of a positive lookup, the list of suffix rules pointed by the hash entry is recovered, and the approach iteratively compares the suffix rules, one by one, to the suffix of the remaining part of the word, i.e., the original word without the last character. The suffix rules comparisons performs likewise the list-based approach.

For example, considering to reduce the word “linguiç” using the hash table from Figure 3, there is a positive lookup for the last character “ç” of the word in

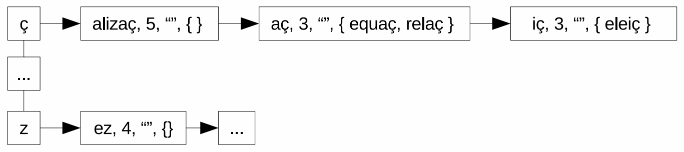

Fig. 3. Example of the data structure for the hash-based approach

the hash table. In this case, the list of suffix rules pointed to the hash entry "ç" is checked. For the first suffix rule, the suffix "alizaç" does not match the suffix of "linguiç", as well as for the second suffix rule. However, for the third suffix rule, "iç" matches the suffix of "linguiç" and, as "linguiç" is not in the exceptions list of the rule and its size is larger than 3, the reduction is performed, reducing "linguiç" to "lingu".

Different from the list-based approach which performs comparisons using all suffix rules, the hash-based approach only performs comparisons using a short list of suffix rules. However, for plural, feminine and degree reductions in Portuguese, the hash-based approach performs likewise the list-based approach, since the hash key entries are composed by only one character, i.e., "s" for plural, "a" for feminine, and "o" for degree. Thus the hash table has only one hash entry pointing to the same list of suffix rules of the list-based approach.

3.3 The Automata-Based Approach

Different from list-based and hash-based approaches that use lists of suffix rules and perform a significantly number of comparisons between rules and word suffixes, the automata-based approach uses a deterministic finite automata (DFA) to reduce the number of comparisons to the minimum. Figure 4 presents an example of a DFA used in noun reductions for words ending in "ç".

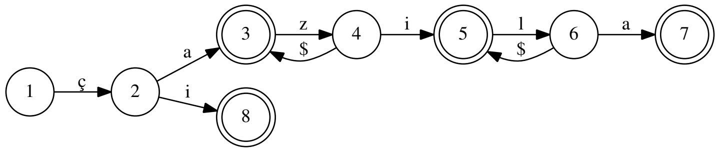

Fig. 4. Example of a DFA used by the automata-based approach

Note that, both word and suffix processing are performed backwards, so we must read the DFA starting by one of the accepting states (states 3, 5, 7 or 8 in Figure 4) and walk until the first state. For instance, if we start in the fifth state and walk until the first state we have the suffix "izaç". In the DFA, it is possible to check if there is a suffix inside another one just checking if an accepting state has a transition to another state. Different from the list-based and hash-based approaches, where suffix rules with larger suffixes must be checked first to avoid wrong reductions, in the automata-based approach the rules with the shorter suffix are checked first. Particularly, we use backtracking to insure the reduction by the longest possible suffix in a way that, if it is impossible to perform the transition for another state just reading a valid character, and there is a state which accepts part of the processed word, a transition to such accepting state is performed. For instance, consider the suffix "aç" (third state), where a suffix satisfying one rule was already found, but it is possible to perform a transition to another state with the character "z", i.e., an attempt to reduce with a longer suffix rule. If the transition to the third state is performed and it is impossible to reach the fourth state, by the backtracking condition, the transition to the third state is performed with the immediate accepting order, so there is no need to perform any character processing. Such accepting order is represented in the DFA with the symbol $ on transitions.

For performance, we consider that there is no need of the transition for the accepting state when there is no backtracking condition for it, e.g., states 7 and 8 in the Figure 4, making possible to perform the reduction at the state that checks the possible transition. So, the state 2 can perform the reduction right after it checks if a transition to state 8 is possible. Thus, we reduce the number of states in the DFA. In addition, we remove the $ transitions, performing rules checking in the states which performs the backtracking. For instance, rules checking can be performed at the states 4 and 6 in Figure 4, reducing the number of DFA transitions. Note that, the automata-based approach performs better when the difference between the size of the suffix rule and the word suffix is larger, resulting in a DFA with few possible state transitions.

3.4 Complexity Analysis

In this section, we present the complexity analysis for the list-based, hash-based and automata-based approaches. In particular, we present the worst case for each approach in terms of space and time. For time complexity, we consider the number of suffix rules that must be processed to reduce (or not) a word, i.e., R in Equation 2. For space complexity, we consider the memory required to store the data structure, i.e., the suffix rules and the hash entries. Orengo and Huyck [7] present the suffix rules used for Portuguese stemming.

The list-based approach compares n suffix rules, one by one, to the word suffix to decide whether a reduction should be performed. The worst case occurs when no reduction is performed and all the suffix rules are compared to the word suffix. Thus, the time complexity is linear. Moreover, the largest value of n occurs in the verb reduction, where $n = 89$.

The hash-based approach compares $m \ll n$ suffix rules, one by one, to the word suffix to decide whether a reduction should be performed. Particularly, the last character of the word is used to lookup the suitable hash entry pointing to the list of suffix rules to be compared.The worst case occurs when the hash entry points to the longest list of rules, comparing the term with all rules of this list.

Thus, the time complexity is sub-linear. The largest value of m occurs in the verb reduction, where the hash entry for words ending in "o" has $m = 25$ rules.

The automata-based approach uses DFA to process the word backwards, and after each character comparison, the suitable reduction is performed by processing a very small number of suffix rules. The worst case occurs by checking the biggest path of the automata to compare $m = 4$ suffix rules for verb reduction. Thus, the time complexity is nearly constant.

The three approaches load all the n suffix rules into memory to use them when necessary, so the space complexity is linear for them. However, while the list-based approach requires space to store n suffix rules in memory, the hash-based approach requires an extra space to store the hash structure, i.e., the entry key and a pointer to the list of rules, and the automata-based approach stores the suffix rules as part of the source code.

4 Experiments

To validate our suffix stripping approaches, we contrast them with a state-of-the-art baseline on two different datasets. Particularly, in this section we answer the following research questions:

1. How efficient are the proposed approaches for suffix stripping?
2. How the proposed approaches perform in each step of suffix stripping?

4.1 Experimental Setup

To assess the efficiency of our suffix stripping approaches, we use the RSLP and WBR99 datasets. The RSLP dataset has been used to validate the effectiveness of Portuguese stemming algorithms [7]. It provides 198 Portuguese words into five categories: noun, verb, plural, feminine, degree (augmentative and diminutive). Table 1 presents the words distribution and the average word size (number of characters) by category in the RSLP dataset. The last column of the Table 1 shows the total number of words and the average word size for the entire dataset.

The WBR99[2] is a standard dataset used in information retrieval [9, 10]. Indeed, it comprises different collections with almost 6 millions Web pages, crawled from the Brazilian Web (the .br domain). In our experiments, we use six different collections from the WBR99, with a vocabulary of more than 206 millions words, excluding numbers and large strings (more than 50 characters). Table 2

Table 1. Words distribution and average sizes by category in the RSLP dataset

	Noun	Verb	Plural	Feminine	Degree	ALL
# of words	64	90	11	15	18	198
Avg. size	4.22	3.55	2.64	3.00	4.94	3.67

Table 2. Words distribution and average sizes by WBR99 collection

Collection	# of words	Avg. word size
AmostRA-NILC	103,874	7.71
CETEMPúblico	201,658,990	9.23
Museu da Pessoa	1,518,881	8.06
ReLi	160,481	7.67
Tycho Brahe	471,504	7.53
Vercial	2,894,625	8.00
ALL	206,808,355	8.03

presents the number of words and the average word size by collection. The last row in Table 2 shows the total of words and the mean average word size.

We contrast the efficiency of our hash-based approach (RSLP-HBA), described in Section 3.2, and automata-based approach (RSLP-ABA), described in Section 3.3, with the state-of-the-art list-based approach baseline (RSLP-LBA), described in Section 3.1. We report efficiency in terms of stemming time, i.e., the average elapsed time in microseconds (μs) for stemming a word, ignoring the time to load the suffix rules and hash entries in memory. The experimental results are averages on 10 trials and significance is verified with a two-tailed paired t-test [5], with the symbol ▲ (▼) denoting a significant increase (decrease) at the $p < 0.01$ level, and the symbol ● denoting no significant difference.

The experiments were carried out on a computer running a Linux kernel version 3.16, with a 64-bit Intel Core i3 2.13 GHz processor, 3 GB of main memory and one SATA II disk of 320 GB. The approaches were implemented using the C++ language.

4.2 Experimental Results

In this section, we present the results of the experiments we have carried out to evaluate our suffix stripping approaches. In particular, we address the two research questions stated previously, contrasting the efficiency of our approaches with the baseline.

Overall Performance In this section, we address our first research question, by assessing the overall efficiency of our approaches on WBR99 collections. To this end, Table 3 shows the efficiency, in terms of stemming time, of each proposed approach and the baseline. Particularly, the percentage improvement compared to the baseline is shown. In addition, a first significance symbol denotes whether the improvements are statistically significant. For the RSLP-ABA, a second such symbol denotes significance with respect to RSLP-HBA. The best value in each column is highlighted in bold.

From Table 3, we observe that the RSLP-ABA significantly improves upon the RSLP-HBA and the RSLP-LBA baseline in all collections. In particular, the

[2] Available at `http://www.linguateca.pt/Repositorio/WBR-99`.

Table 3. Stemming time (μs) for suffix stripping on WBR99 collections

Collection	RSLP-LBA	RSLP-HBA	RSLP-ABA
AmostRA-NILC	1.37346	0.57674 (58.00%) ▼	**0.20779 (84.87%) ▼▼**
CETEMPúblico	0.03657	0.01338 (63.41%) ▼	**0.00504 (86.21%) ▼▼**
Museu da Pessoa	0.23419	0.09632 (58.87%) ▼	**0.03729 (84.07%) ▼▼**
ReLi	0.75644	0.31680 (58.11%) ▼	**0.12272 (83.77%) ▼▼**
Tycho Brahe	0.53799	0.19651 (63.47%) ▼	**0.07759 (85.57%) ▼▼**
Vercial	0.18901	0.06561 (65.28%) ▼	**0.02555 (86.48%) ▼▼**
All	0.52128	0.21090 (59.54%) ▼	**0.07933 (84.78%) ▼▼**

gains are up to 86.48%. Additionally, we also observe that RSLP-HBA outperforms the baseline in all collections with gains of up to 65.28%. Recalling our first research question, these observations attest the efficiency of our approaches for suffix stripping.

Performance by Category In this section, we address our second research question, by assessing the efficiency of our approaches in different categories: noun, verb, plural, feminine, and degree. The aforementioned categories are the same categories in the RSLP dataset presented in Section 4.1, and also corresponds to the suffix stripping steps described in Section 3, except for the step 4, which is not considered, since the adverb reduction has only one suffix rule.

To this end, Table 4 shows the efficiency by category, in terms of stemming time, of each proposed approach and the baseline. Particularly, the percentage improvement compared to the baseline is shown. In addition, a first significance symbol denotes whether the improvements are statistically significant. For the RSLP-ABA, a second such symbol denotes significance with respect to RSLP-HBA. The best value in each column is highlighted in bold.

Table 4. Stemming time (μs) for suffix stripping by category on the RSLP dataset

Category (Step)	RSLP-LBA	RSLP-HBA	RSLP-ABA
Noun	2.20733	1.07529 (51.28%) ▼	**0.46008 (79.15%) ▼▼**
Verb	3.63152	1.54178 (57.54%) ▼	**0.68953 (81.01%) ▼▼**
Plural	**0.16546**	**0.16546 (00.00%) ●**	0.21505 (-29.97%) ▲▲
Feminine	0.28770	0.28770 (00.00%) ●	**0.20967 (27.12%) ▼▼**
Degree	0.22786	0.22786 (00.00%) ●	**0.18010 (19.64%) ▼▼**

From Table 4, we observe that for all categories, except plural, the RSLP-ABA significantly improves upon the RSLP-HBA and the RSLP-LBA baseline. In particular, the gains are up to 81.01%. Additionally, we also observe that RSLP-HBA outperforms the baseline for noun and verb with gains of up to 57.54%. Note that, for plural, feminine and degree, the RSLP-LBA and RSLP-HBA performs equally, since the suffix used to distribute suffix rules by hash entries, for each one of these categories, are composed by only one character, i.e., “s” for plural, “a” for feminine, and “o” for degree. Thus the hash table has only one hash entry pointing to the same list of suffix rules of the RSLP-LBA.

As mentioned is Section 3.3, the RSLP-ABA performs better when the difference between the size of the suffix rule and the word suffix is larger, resulting in a DFA with few possibilities of state transitions. From Table 4, we observe that RSLP-ABA presents greater gains for noun and verb than for feminine and degree. This performance gap occurs because for noun and verb, the difference between the size of the suffix rule and the word suffix is larger than for feminine and degree. Moreover, for plural, where the suffix rule and the word suffix have the same size, the RSLP-ABA uses a compact DFA with several possibilities of state transitions, resulting on the only case reported in Table 4 in which RSLP-LBA outperforms RSLP-ABA. Note that, plural reduction occurs in 10.60% of the total amount of reductions performed in our experiments. In summary, the RSLP-ABA outperforms the RSLP-LBA baseline for all categories, except plural. In addition, RSLP-HBA outperforms the RSLP-LBA baseline for noun and verb and presents the same performance for plural, feminine and degree. Recalling our second research question, these observations attest the efficiency of our approaches in each step of suffix stripping.

5 Conclusion

In this article we introduce two novel suffix stripping approaches for Portuguese stemming, the hash-based and the automata-based approaches. We also assess the efficiency of these approaches contrasting them with a state-of-the-art baseline from the literature, the list-based approach. Particularly, the list-based and hash-based approaches iteratively process lists of suffix rules to reduce a word to a stem. However, the list-based approach uses a single list of suffix rules while the hash-based approach uses multiple lists of rules grouped by word suffixes of the language. Different from the other two approaches, the automata-based approach uses a DFA to iteratively compare suffix characters to decide when to perform (or not) a reduction, avoiding unnecessary suffix rules processing, frequent in the other two approaches.

The proposed approaches are simple, effective and can be adapted to work with different languages, such as English and Spanish. In addition, the complexity analysis shows that the list-based approach presents linear complexity both in space and time on the number of suffix rules to process, while the hash-based approach presents linear complexity in space and sub-linear complexity in time, and the automata-based approach presents linear complexity in space and constant complexity in time. Moreover, we thoroughly evaluated the approaches using the RSLP and WBR99 datasets. The results of this evaluation attest the efficiency of the automata-based approach, with gains of up to 86.48% in stemming time over the baseline. By breaking down our analysis by categories, we demonstrated the robustness of the automata-based approach to perform reductions on noun, verb, feminine and degree words, with gains of up to 81.01% in stemming time.

For future work, we plan to deploy and assess the efficiency of a distributed version of the proposed approaches. Another plan is to adapt the proposed approaches to work with other languages.

Acknowledgements

We thank the partial support given by PUC Minas (grant FIP 2015/9396-S1), and CNPq Grant 444156/2014-3 (Wladmir C. Brandão).

References

1. Reinaldo Viana Alvares, Ana Cristina Bicharra Garcia, and Inhaúma Ferraz. Stembr: A stemming algorithm for the brazilian portuguese language. In *Progress in Artificial Intelligence*, volume 3808 of *Lecture Notes in Computer Science*, pages 693–701. Springer Berlin Heidelberg, 2005.
2. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval: the concepts and technology behind search.* Pearson Education, Harlow, England, 2nd edition, 2011.
3. Alexandre Ramos Coelho. *Stemming for the Portuguese language: study, analysis and improvement of the RSLP algorithm.* Monography, Universidade Federal do Rio Grande do Sul, 2007.
4. Matt Cutts. Spotlight keynote. In *Proceedings of Search Engines Strategies*, San Francisco, CA, USA, 2012.
5. Raj Jain. *The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling.* Wiley-Interscience, New York, 1991.
6. Viviane Moreira Orengo, Luciana Salete Buriol, and Alexandre Ramos Coelho. A study on the use of stemming for monolingual ad-hoc portuguese information retrieval. In *Evaluation of Multilingual and Multi-modal Information Retrieval*, volume 4730 of *Lecture Notes in Computer Science*, pages 91–98. Springer Berlin Heidelberg, 2007.
7. Viviane Moreira Orengo and Christian Huyck. A stemming algorithm for the portuguese language. In *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE)*, pages 186–193, 2001.
8. Martin F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 40:211–218, 2006.
9. Bruno Pôssas, Nivio Ziviani, Wagner Meira Jr., and Berthier A. Ribeiro-Neto. Set-based vector model: An efficient approach for correlation-based ranking. *ACM Transactions on Information Systems*, 23(4):397–429, 2005.
10. Bruno Pôssas, Nivio Ziviani, Berthier A. Ribeiro-Neto, and Wagner Meira Jr. Processing conjunctive and phrase queries with the set-based model. In *Proceedings of the 11th International Conference on String Processing and Information Retrieval (SPIRE)*, pages 171–182, 2004.