University of Padua at CLEF 2002: Experiments to evaluate a statistical stemming algorithm

Michela Bacchin Nicola Ferro Massimo Melucci

{michela.bacchin, nicola.ferro, massimo.melucci}@unipd.it

Department of Information Engineering – University of Padua
Via Gradenigo 6/a – 35131 Padova, Italy

Abstract

In Information Retrieval (IR), stemming is used to reduce variant word forms to common root. The assumption is that if two words have the same root, then they represent the same concept. Hence stemming permits a IR system to match query and document terms which are related to a same meaning but which can appear in different morphological variants. In this paper we will report our participation in CLEF 2002 Italian monolingual task, whose aim was to evaluate a statistical stemming algorithm based on link analysis. Considering that a word is formed by a prefix (stem) and a suffix, the key idea is that the interlinked prefixes and suffixes form a community of substrings. Hence discovering these communities means searching for the best word splits which give the best word stems. The results show that stemming improves the IR effectiveness. They also show that effectiveness level of our algorithm, which does not incorporate any heuristics nor linguistic knowledge, is comparable to that of an algorithm based on a-priori linguistic knowledge. This is an encouraging result, particularly in a multi-lingual context.

1 Introduction

The main objective of our current research in multi-lingual Information Retrieval is to design, implement and evaluate a language-independent stemming algorithm based on statistical methods. We participated in CLEF 2002 Italian monolingual task with the aim of evaluating a statistical algorithm based on link-analysis methods. To accomplish this objectives we designed our experiments to investigate if stemming does not deteriorate or even enhances the effectiveness of retrieval of documents written in Italian by making use of both a linguistic stemmer and a statistical stemmer. Then we investigated if a statistical and language-independent stemming algorithm can perform as effectively as an algorithm developed on the basis of a-priori linguistic knowledge. This paper reports our participation in CLEF 2002 and it is organized as follows: Section 2 introduces the stemming process, section 3 reports the methodological approach we followed to build a new statistical and language-independent stemming algorithm, section 4 describes our official runs and the results we obtained, and section 5 reports some conclusions and future work.

2 Stemming

Stemming is used to reduce variant word forms to common root. The assumption is that if two words have the same root, then they represent the same concept. Hence stemming permits a IR system to match query and document terms which are related to a same meaning but which can appear in different morphological variants.

The effectiveness of stemming is a debated issue, and there are different results and conclusions. If effectiveness is measured by the traditional precision and recall measures, it seems that for a language with a relatively simple morphology, like English, stemming influences the overall performance little [6]. In contrast, stemming can significantly increase the retrieval effectiveness [13] and can also increase precision for short queries, [8] for languages with a more complex morphology, like the romance languages. Finally, as the system performance must reflect user's expectations it has to be considered that the use of a stemmer is intuitive to many users [6], who can express the query to the system using a specific word without keeping in mind that only a variant of this word can appear in a relevant document. Hence, stemming can be viewed also as a sort of feature related to the user-interaction interface of an IR service.

To design a stemming algorithm, it is possible to follow a linguistic approach, using prior knowledge of the morphology of the specific language, or a statistical approach using some methods based on statistical principles to infer from the corpus of documents the word formation rules in the language studied. The former implies manual labor which has to be done by experts in linguistics – as matter of the fact, it is necessary to formalize the word formation rules, the latter being hard work, especially for those languages whose morphology is complex. Stemming algorithms based on statistical methods ensure no costs for inserting new languages on the system, and this is an advantage that becomes crucial especially for multilingual IR systems.

3 Methodological Approach

We will consider a special case of stemming, which belongs to the category known as *affix removal stemming* [3]. In particular our approach stays on a suffix stripping paradigm which is adopted by most stemmers currently in use by IR, like those reported in [9, 12, 15]. This stemming process splits each word into two parts, prefix and suffix, and considers the stem as the substring corresponding to the obtained prefix. Let us consider a finite collection of unique words $W = \{w_1, ..., w_N\}$ and a word $w \in W$ of length $|w|$, then w can be written as $w = xy$ where x is a prefix and y is a suffix. If we split each word w into all the $|w| - 1$ possible pairs of substrings, we build a collection of substrings, and each substring may be either a prefix, a suffix or both of at least an element $w \in W$. Let X be the set of the prefixes of the collection and $S \subseteq X$ be the set of the stems. We are interested in detecting the prefix x that is the most probable stem for the observed word w. Hence, we have to determine the prefix x^* such as:

$$x^* = \arg\max_x Pr(x \in S \mid w \in W) \quad (1)$$

$$= \arg\max_x \frac{Pr(w \in W \mid x \in S) Pr(x \in S)}{Pr(w \in W)} \quad (2)$$

where (2) is obtained applying the Bayes' theorem which lets us swap the order of dependence between events. We can ignore the denominator, which is the same for all splits of w. $Pr(w \in W \mid x \in S)$ is the probability of observing w given that the stem x has been observed. A reasonable estimation of that probability would be the reciprocal of the number of words beginning by that stem if the stems were known. However note that the stems are unknown – indeed stem detection is the target of this method – and the number of words beginning by a stem cannot be computed. Therefore we estimated that probability by the reciprocal of the number of words beginning by that prefix. As regards $Pr(x \in S)$ we estimated this probability using an algorithm that discloses the mutual relationship between stems and derivations in forming the words of the collection.

The rationale of using mutual reinforcement is based on the idea that stems extracted from W are those substrings that:

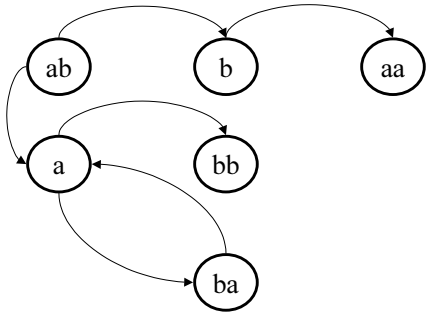
- are very frequent, and
- form words together with very frequent suffixes.

This means that very frequent prefixes are candidate to be stems, but they are discarded if they are not followed by very frequent suffixes; for example, all initials are very frequent prefixes but they are

unlikely stems because the corresponding suffixes are rather rare, if not unique – the same holds for suffixes corresponding to ending vowels or consonants. Thus, there are prefixes being less frequent than initials, but followed by frequent suffixes, yet less frequent than ending characters: these suffixes and prefixes correspond to candidate correct word splits and we label them as "good". The key idea is that interlinked good prefixes and suffixes form a community of substrings whose links correspond to words, i.e. to splits. Discovering these communities is like searching for the best splits.

To compute the best split, we used the quite well-known algorithm called HITS reported in [7] and often discussed in many research papers as a paradigmatic algorithm for Web page retrieval. It considers a mutually reinforcing relationship among good authorities and good hubs, where an authority is a web page pointed to by many hubs and a hub is a web page which points to many authorities. The parallel with our context will be clear when we associate the concept of a hub to a prefix and that of authority to a suffix. The method belongs to the larger class of approaches based on frequencies of substrings to decide the goodness of prefixes and suffixes, often used in statistical morphological analysis [11, 4], and in the pioneer work [5]. The contribution of this paper is the use of mutual reinforcement notion applied to prefix frequencies and suffix frequencies, to compute the best word splits which give the best word stems as explained in the following.

Using a graphical notation, the set of prefixes and suffixes can be written as a graph g such that nodes are substrings and an edge occurs between nodes x, y if $w = xy$ is a word in W. By definition of g, no vertex is isolated. As an example, let us consider the following toy set of words: W={`aba`, `abb`, `baa`}; splitting these into all the possible prefixes and suffixes produces a graph, reported in Figure 3a.



(a)

substring	prefix score	suffix score
`a`	0.250	0.333
`aa`	0.000	0.167
`ab`	0.375	0.000
`b`	0.125	0.167
`ba`	0.250	0.167
`bb`	0.000	0.167

(b)

Figure 1: (a) The graph obtained from W. (b) The prefix and suffix scores from W

If a directed edge exists between x and y, the mutual reinforcement notion can be stated as follows:

good prefixes point to good suffixes, and good suffixes are pointed to by good prefixes.

Let us define $P(y) = \{x : \exists w, w = xy\}$ and $S(x) = \{y : \exists w, w = xy\}$ that are, respectively, the set of all prefixes of a given suffix y and the set of all suffixes of a given prefix x. If p_x and s_x indicate, respectively, the prefix score and the suffix score, the criteria can be expressed as:

$$p_x = \sum_{y \in S(x)} s_y \qquad s_y = \sum_{x \in P(y)} p_x \tag{3}$$

under the assumption that scores are expressed as sums of scores and splits are equally weighed.

The method of mutual reinforcement has been formalized through the HITS iterative algorithm. Here we map HITS in our study context, as follows:

Compute suffix scores and prefix scores from W
V: the set of substrings extracted from all the words in W

$P(y)$: the set of all prefixes of a given suffix y
$S(x)$: the set of all suffixes of a given prefix x
N: the number of all substrings in V
n: the number of iterations
$\mathbf{1}$: the vector $(1, ..., 1) \in \mathcal{R}^{|V|}$
$\mathbf{0}$: the vector $(0, ..., 0) \in \mathcal{R}^{|V|}$
$\mathbf{s}^{(k)}$: suffix score vector at step k
$\mathbf{p}^{(k)}$: prefix score vector at step k
$\mathbf{s}^{(0)} = \mathbf{1}$
$\mathbf{p}^{(0)} = \mathbf{1}$
for each iteration $k = 1, ..., n$
$\mathbf{s}^{(k)} = \mathbf{0}$
$\mathbf{p}^{(k)} = \mathbf{0}$
for each $y \in V$
$s_y^{(k)} = \sum_{x \in P(y)} p_x^{(k-1)}$;
for each $x \in V$
$p_x^{(k)} = \sum_{y \in S(x)} s_y^{(k)}$;
normalize $\mathbf{p}^{(k)}$ and $\mathbf{s}^{(k)}$ so that $1 = \sum_x p_x^{(k)} = \sum_y s_y^{(k)}$
end.

Using the matrix notation, the graph g can be described with a $|V| \times |V|$ matrix $\mathbf{M}$ such that

$$m_{ij} = \begin{cases} 1 & \text{if prefix } i \text{ and suffix } j \text{ form a word} \\ 0 & \text{otherwise} \end{cases}$$

As explained in [7], the algorithm computes two matrices: $\mathbf{A} = \mathbf{M}^T\mathbf{M}$ and $\mathbf{B} = \mathbf{M}\mathbf{M}^T$, where the generic element a_{ij} of $\mathbf{A}$ is the number of vertices that are pointed by both i and j, whereas the generic element b_{ij} of $\mathbf{B}$ is the number of vertices that point to both i and j. The n-step iteration of the algorithm corresponds to computing $\mathbf{A}^n$ and $\mathbf{B}^n$. In the same paper, it has been argued that $\mathbf{s} = [s_y]$ and $\mathbf{p} = [p_x]$ converge to the eigenvectors of $\mathbf{A}$ and $\mathbf{B}$, respectively. The scores computed for the toy set of words are reported in Table 3b.

As explained previously, we argue that the probability that x is a stem, can be estimated with the prefix score p_x just calculated. The underlying assumption is that the scores can be seen as probabilities, and, in effect, it has been proved in a recent work that HITS scores can be considered as a stationary distribution of a random walk [1]. In particular, the authors proved the existence of a Markov chain, which has the stationary distribution equal to the hub vector after the n^{th} iteration of the Kleinberg's algorithm, which is, in our context, the prefix score vector $\mathbf{p} = [p_x]$. The generic element $q_{ij}^{(n)}$ of the transition matrix referred to the chain is the probability that, starting from i, one reaches j after n "bouncing" to one of the suffixes which begins to be associated with i and j. To interpret the result in a linguistic framework, p_i can be seen as the probability that i is judged as a stem by the same community of substrings (suffixes) being resulted by the process of splitting words of a language. In Table 1, all the possible splits for all the words are reported and measured using the estimated probability.

4 Experiments

The aim of CLEF 2002 experiments is to compare the retrieval effectiveness of the link analysis-based algorithm illustrated in the previous Section with that of an algorithm based on a-priori linguistic knowledge, because the hypothesis is that a language-independent algorithm, such as the one we propose, might effectively replace one developed on the basis of manually coded derivational rules. Before comparing the algorithms, we assessed the impact of both stemming algorithms by comparing their effectiveness with that reached without any stemmer. In fact, we did want to test if the system performance is not

word	prefix	suffix	words beginning by prefix	words ending by suffix	probability	choice
baa	b	aa	1	1	0.1250	
baa	ba	a	1	2	0.2500	*
aba	a	ba	2	1	0.1250	
aba	ab	a	2	2	0.1875	*
abb	a	bb	2	1	0.1250	
abb	ab	b	2	1	0.1875	*

Table 1: The candidate splits from W={aba, baa, abb}.

significantly hurt by the application of stemming, as hypothesized in [6]. If stemming did on the contrary improve effectiveness, and the effectiveness of the tested algorithms were comparable, the link-based algorithm would ensure low costs for extending it also to other languages, which is crucial in multi-lingual settings. To evaluate stemming, we decided to compare the performance of an IR system changing only the stemming algorithms for different runs, all other things being equal.

4.1 Experimental Prototype System

For indexing and retrieval, we used an experimental IR system, called IRON, which has been realized by our research group with the aim of having a robust tool for carrying out IR experiments. IRON is built on top of the Lucene 1.2 RC4 library, which is an open-source library for IR written in Java and publicly available in [10]. The system implements the vector space model [16], and a $(\text{tf} \cdot \text{idf})$–based weighting scheme [17]. The stop-list which was used consists of 409 Italian frequent words and it is publicly available in [18].

As regards the realization of the statistical stemming algorithm, we built a suite of tools, called Stemming Program for Language Independent Tasks (SPLIT), which implements the link-based algorithm and chooses the best stem, according to the probabilistic criterion described in Section 3. From the vocabulary of the Italian CLEF sub-collection, SPLIT spawns a 2,277,297-node and 1,215,326-edge graph, which is processed to compute prefix and suffix scores – SPLIT took 2.5 hours for 100 iterations on a personal computer equipped with Linux, an 800 MHz Intel CPU and 256MB RAM.

4.2 Runs

We tested four different stemming algorithms:

1. **NoStem**: No stemming algorithm was applied.

2. **Porter-like**: We used the stemming algorithm for the Italian language, which is freely available in the Snowball Web Site [14] edited by M. Porter. Besides being publicly available for research purposes, we have chosen this algorithm because it uses a kind of a-priori knowledge of the Italian language, so comparing our SPLIT algorithm with this particular "linguistic" algorithm could give some information about the possibility of estimating linguistic knowledge with statistically inferred knowledge.

3. **SPLIT**: We implemented our first version of the stemming algorithm based on a link-analysis with 100 iterations.

4. **SPLIT-L3**: We included in our stemming algorithm a little ignition of linguistic knowledge, inserting a heuristic rule which forces the length of the stem to be at least 3.

4.3 A Global Evaluation

We carried out a macro evaluation by averaging the results over all the queries of the test collection. Table 2 shows a summary of the figures related to the macro analysis of the stemming algorithm for 2002 topics, while table 3 reports 2001 data.

Run ID	Algorithm	N. Relevant Retrieved	Av. Precision	R-Precision
PDDN	NoStem	887	0.3193	0.3367
PDDP	Porter-like	914	0.3419	0.3579
PDDS2PL	SPLIT	913	0.3173	0.3310
PDDS2PL3	SPLIT-L3	911	0.3200	0.3254

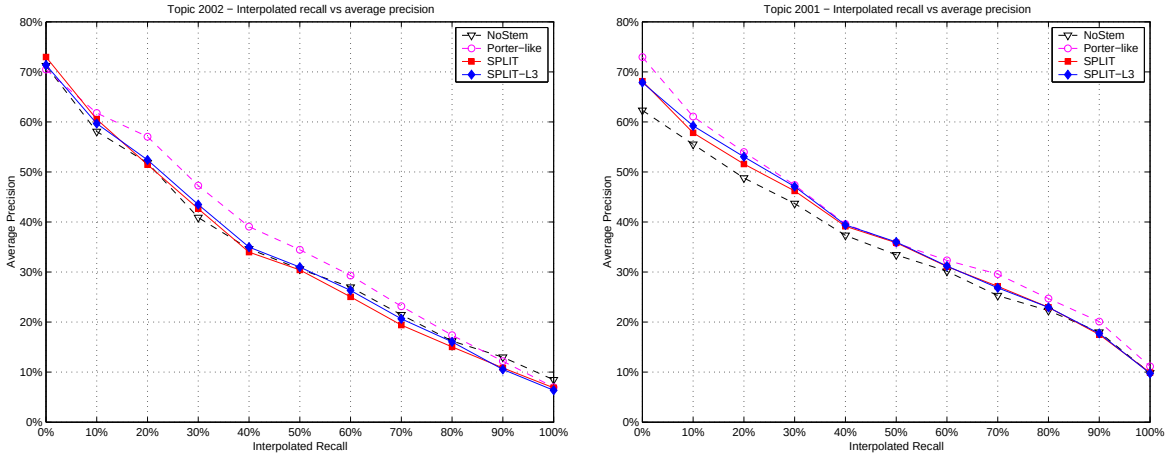
Table 2: Macro comparison among runs for 2002 topics.

Algorithm	N. Relevant Retrieved	Av. Precision	R-Precision
NoStem	1093	0.3387	0.3437
Porter-like	1169	0.3753	0.3619
SPLIT	1143	0.3519	0.3594
SPLIT-L3	1149	0.3589	0.3668

Table 3: Macro comparison among runs for 2001 topics.

Note that both for 2001 and 2002 topics, all the considered stemming algorithms improve recall, since the number of retrieved relevant documents is larger than the number of retrieved relevant documents observed in the case of retrieval without any stemmer; the increase has been observed for all the stemming algorithms. As regards the precision, while for 2002 topics, stemming does not hurt the overall performances of the system, for 2001 data, stemming even increases the precision, and then the overall performance is higher thanks to the application of stemming.

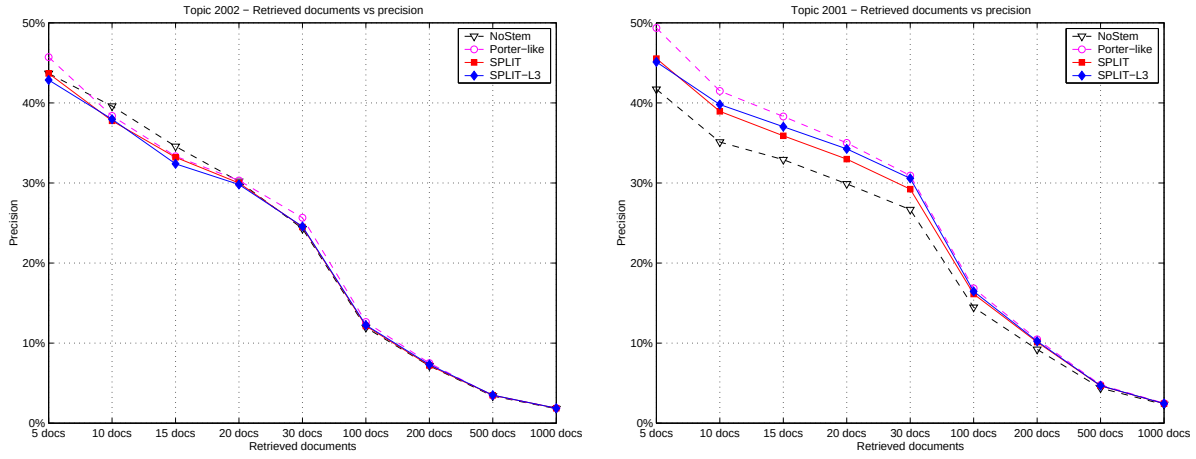
Figure 2 shows the Averaged Recall-Precision curve at different levels of recall and Figure 3 illustrates the Recall-Precision curve at given document cutoff values, both for 2002 and 2001 topic sets. As regards



(a) 2002 Topics (b) 2001 Topics

Figure 2: Average Precision Curves for four stemming algorithms.

the use of link-based stemming algorithms, it is worth noting that SPLIT can attain levels of effectiveness being comparable to one based on linguistic knowledge. This is surprising if you know that SPLIT was



(a) 2002 Topics

(b) 2001 Topics

Figure 3: R-Precision Curves for four stemming algorithms.

built without any sophisticated extension to HITS and that neither heuristics nor linguistic knowledge was used to improve effectiveness. It should also be considered as a good result, if you consider that it has also been obtained for the Italian language, which is morphologically more complex than English.

5 Conclusions and Future Work

The objective of this research was to investigate a stemming algorithm based on link analysis procedures. The idea has been that prefixes and suffixes, that are stems and derivations, form communities once extracted from words. We tested this hypothesis by comparing the retrieval effectiveness of SPLIT, a link analysis based algorithm derived from HITS, with a linguistic knowledge based algorithm, on a quite morphologically complex language as it is the Italian language.

The results are encouraging because effectiveness level of SPLIT is comparable to that developed by Porter. The results should be considered even better since SPLIT does not incorporate any heuristics nor linguistic knowledge. Moreover, stemming, and then SPLIT, showed to improve effectiveness with respects to not using any stemmer.

We are carrying out further analysis at a micro level to understand the conditions under which SPLIT performs better or worse compared to other algorithms. Further work is in progress to improve the probabilistic decision criterion and to insert linguistic knowledge directly in the link-based model by thus weighting links among prefixes and suffixes with a probabilistic function which could capture available information on the language, such as, for example, the minimum length of a stem. Finally, further experimental work is in progress with other languages.

References

[1] A. Borodin, G.O. Roberts, J.S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the World Wide Web. In *Proceedings of the World Wide Web Conference*, pages 415–429, Hong Kong, 2001. ACM Press.

[2] C. Cleverdon. The Cranfield Tests on Index Language Devices. In K. Sparck Jones and P. Willett (Eds.). *Readings in Information Retrieval*, pages 47-59, Morgan Kaufmann, 1997.

[3] W.B. Frakes and R. Baeza-Yates. *Information Retrieval: data structures and algorithms*. Prentice Hall, 1992.

[4] J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):154–198, 2001.

[5] M. Hafer and S. Weiss. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385, 1994.

[6] D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7–15, 1991.

[7] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.

[8] R. Krovetz. Viewing Morphology as an Inference Process,. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 1993.

[9] J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[10] The Jakarta Project. Lucene. `http://jakarta.apache.org/lucene/docs/index.html`, 2002.

[11] C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. The MIT Press, 1999.

[12] C.D. Paice. Another Stemmer. In *ACM SIGIR Forum*, 24, 56–61, 1990.

[13] M. Popovic and P. Willett. The effectiveness of stemming for natural-language access to sloven textual data. *Journal of the American Society for Information Science*, 43(5):383–390, 1992.

[14] M. Porter. Snowball: A language for stemming algorithms. `http://snowball.sourceforge.net`, 2001.

[15] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[16] G. Salton and M. McGill. *Introduction to modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.

[17] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[18] Institut interfacultaire d'informatique. CLEF and Multilingual information retrieval. University of Neuchatel. `http://www.unine.ch/info/clef/`, 2002.

[19] C. Buckley. Trec_eval. `ftp://ftp.cs.cornell.edu/pub/smart/`, 2002.