The Effectiveness of a Graph-Based Algorithm for Stemming

Michela Bacchin, Nicola Ferro, and Massimo Melucci

Department of Information Engineering
University of Padua,
Via Gradenigo, 6/a – 35031 Padova – Italy
{michela.bacchin, nicola.ferro, massimo.melucci}@unipd.it

Abstract. In Information Retrieval (IR), stemming enables a matching of query and document terms which are related to a same meaning but which can appear in different morphological variants. In this paper we will propose and evaluate a statistical graph-based algorithm for stemming. Considering that a word is formed by a stem (prefix) and a derivation (suffix), the key idea is that strongly interlinked prefixes and suffixes form a community of sub-strings. Discovering these communities means searching for the best word splits which give the best word stems. We conducted some experiments on CLEF 2001 test sub-collections for Italian language. The results show that stemming improve the IR effectiveness. They also show that effectiveness level of our algorithm 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

In an Information Retrieval (IR) system that manages text resources, indexing is the process that assigns a set of best content describing index terms to each document or query. Usually, both documents and queries are written in natural language, so the words may often occur with many morphological variants, even if they are referred to as a common concept. The basic idea which exists in stemming is that words which are similar in morphology are likely to be similar in meaning, so they can be considered as equivalent from an IR point of view. The goal of a stemming algorithm is to reduce variant word forms to a common morphological root, called "stem".

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,[1] it seems that for a language with a relatively simple morphology, like English, stemming influences the overall performance little. [8] In contrast, stemming can significantly increase the retrieval effectiveness and can

[1] *Recall* is the fraction of relevant documents that has been retrieved, and *precision* as the fraction of retrieved documents that are relevant.

also increase precision for short queries or for languages with a more complex morphology, like the romance languages. [10,15] 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, who can express the query to the system using a specific term without keeping in mind that only a variant of this term can appear in a relevant document. [8] 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 kind of algorithms imply 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 applications to Digital Libraries which are often constructed for a particular institution or nation, and can manage a great amount of non-English documents as well as documents written in more than one different languages.

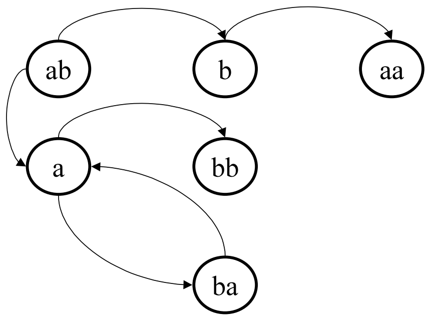
2 Methodological Approach

We will consider a special case of stemming, which belongs to the category known as *affix removal stemming.* [5] 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 [11,14,18]. This stemming process splits each word into two parts, prefix and suffix, and considers the stem as the sub-string corresponding to the obtained prefix.

By exploiting a sort of mutual reinforcement between prefix and suffix of a word, we compute the best stem and the best derivation, i.e. the best split, of the word. The rationale of using mutual reinforcement is based on the idea that stems extracted from a finite collection of unique words are those prefixes that are very frequent and form words together with very frequent suffixes. The key idea is that interlinked good prefixes and suffixes form a community of sub-strings whose links correspond to words, i.e. to splits. Discovering these communities is like searching for the best splits. Note 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”.

2.1 Mutual Reinforcement as a Method for Discovering the Best Stems

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, provided $|x| > 0$ and $|y| > 0$. If we split each word w into all the $|w|-1$ possible pairs of sub-strings, we build a collection of sub-strings, and each sub-string may be either a prefix, a suffix or both of at least an element $w \in W$. Using a graphical notation, the set of prefixes and suffixes can be written as a graph $g = (V, E)$ such that V is the set of sub-strings and $w = (x, y) \in E$ is an edge w that 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 2.1a, with vertex set V={`a`, `b`, `aa`, `ba`, `ab`, `bb`} and edge set {(`ab`,`a`), (`ba`,`a`), (`b`,`aa`), (`ab`,`b`), (`a`,`ba`), (`a`,`bb`)}.



sub-string	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

(a) (b)

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

The method used to compute the best split of each word employs the notion of mutual reinforcement and the criteria based on frequencies of sub-strings to decide the goodness of prefixes and suffixes, often used in statistical morphological analysis, [13,6] and in the pioneer work. [7] 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.

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.

In mathematical form, 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 is the prefix score, i.e. the degree to which the prefix x is a stem, and s_y is the suffix score, i.e. the degree to which the suffix y is a derivation, then the mutual reinforcing relationship can be expressed as:

$$s_y = \sum_{x \in P(y)} p_x \qquad p_x = \sum_{y \in S(x)} s_y \tag{1}$$

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

2.2 The Estimation of Prefix Scores

To estimate the prefix score, we used the quite well-known algorithm called HITS (Hyper-link Induced Topic Search) reported in [9] 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.

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

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

As explained in [9], the algorithm computes two matrices after the first iteration: $\mathbf{A} = \mathbf{M}^T\mathbf{M}$ 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 k-step iteration of the algorithm corresponds to computing $\mathbf{A}^k$ and $\mathbf{B}^k$. In the same paper, it has been argued that $\mathbf{s} = [s_j]$ and $\mathbf{p} = [p_i]$ converge to the eigenvectors of $\mathbf{A}$ and $\mathbf{B}$, respectively.

Here we map HITS in our study context, as follows:

```
Compute suffix scores and prefix scores from $W$
$V$: the set of sub-strings extracted from all the words in $W$
$N$: the number of all sub-strings 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 $k = 1, ..., n$
        $\mathbf{s}^{(k)} = \mathbf{0}$
        $\mathbf{p}^{(k)} = \mathbf{0}$
        for each $y$
                $s_y^{(k)} = \sum_{x \in P(y)} p_x^{(k-1)}$;
        for each $x$
                $p_x^{(k)} = \sum_{y \in S(x)} s_y^{(k)}$;
        normalize $p^{(k)}$ and $s^{(k)}$ so that $1 = \sum_i p_i^{(k)} = \sum_j s_j^{(k)}$
for each $x$
        $p_x^{(n)} = p_x^{(n)} / |S(x)|$
end.
```

Differently from HITS, each prefix score p_x is divided after the n-th iteration by the number of words with the prefix x, i.e. the number of out-links of the node corresponding to the prefix x. The latter arithmetic operation provides an estimation of the probability that x is a stem of a given word. This probability is a component of a probabilistic framework, see [1] for a more detailed discussion, since the illustration of this framework is out of the scope of this paper. However, we explain why the scores can be modeled within a probabilistic framework. In a recent work, it has been proved that HITS scores can be considered as a stationary distribution of a random walk. [2] In particular, it has been proved the existence of a Markov chain $M^{(k)}$, which has the stationary distribution equal to the hub vector after the k^{th} iteration of the Kleinberg's algorithm, which is, in our context, the prefix score vector $\mathbf{p} = [p_j]$. The generic element $q_{ij}^{(k)}$ of the transition matrix referred to $M^{(k)}$ is the probability that, starting from i, one reaches j after k "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 sub-strings (suffixes) being resulted by the process of splitting words of a language. Considering scores as probabilities permits us to model our graph-based stemming algorithm within a probabilistic framework [1].

In Table 1, all the possible splits for all the words are reported and measured using the estimated probability. For each word we choose as stem the prefix with the highest probability.

3 Experiments

The aim of the 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

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

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

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 also want to test if the system performance is not significantly hurt by the application of stemming, as hypothesized in [8]. 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. We conducted the evaluation procedure following the trusted Cranfield methodology, [4] which requires us to evaluate an IR system on a test collection consisting of a set of documents, a set of queries and a list of relevance judgments – each judgment states whether a judged document is relevant or not for each query.

3.1 Experimental Setting

We carried out the retrieval experiments by using a test collection, an experimental prototype system, a suite of effectiveness measures for reflecting the search quality, and statistical methods for judging whether differences between runs can be considered statistically significant.

Test Collection. We carried out the retrieval experiments on the Italian subcollections of the Cross-Language Evaluation Forum (CLEF) 2001 test collection. CLEF is a series of evaluation campaigns which has been held once a year since 1999. [3,16] It offers an infrastructure for testing, tuning and evaluating IR systems operating on European languages. The test documents consist of two distinct subsets of articles both referring to year 1994:

– *La Stampa*, which is an Italian national newspaper;
– Italian SDA, which is the Italian portion of the news-wire articles of SDA (Swiss Press Agency).

We want to concentrate on a morphologically complex European language, as it is the Italian language, because it poses new challenges to stemming which is what we want to investigate. Main features of the test collection are reported in Table 2. After a simple case normalization, the Italian sub-collection has a vocabulary of 333,828 unique words. The query set consists of 50 topics, each one described by a *Title*, a *Description* and a body called *Narrative.*

Table 2. Main features of the collection used in the experiments.

	La Stampa	SDA	Total
Size in KB	198,112	87,592	285,704
Number of documents	58,051	50,527	108,578

Experimental 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 [12]. The system implements the vector space model, [19] and a $(\mathrm{tf} \cdot \mathrm{idf})$–based weighting scheme. [20] The stop-list which was used consists of 409 Italian frequent words and it is publicly available in [21].

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 graph-based algorithm described in Section 2. Using the vocabulary extracted from 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.

Effectiveness Measures. We used R-precision, which is the precision after R relevant retrieved documents, and Average Precision, computed by the trusted evaluation program `trec_eval` developed as part of the experimental SMART system at Cornell University and freely available from [22].

To test the statistical significance of the difference between the compared runs, we carried out a statistical analysis considering each query as a statistical unit and applying the paired Wilcoxon test, which is a non-parametric statistical test working as follows: given two lists X, Y of measures – one list of each run observed – that test replaces each difference D_i between a pair of measures X_i, Y_i with the rank of its absolute value multiplied by the difference sign. The statistics is then $\sum R_i \ / \ \sqrt{\sum R_i^2}$, where $R_i = sign(D_i) \times rank|D_i|$ and is compared to its expected value under the null hypothesis that lists are equal.

3.2 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 edited by M. Porter. [17] 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.
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.

3.3 A Global Evaluation

We carried out a macro evaluation by averaging the results over all the queries of the test collection. Out the 50 queries of the test collection, 47 queries have relevant documents in the collection, so only these queries were evaluated in the analysis. Table 3 shows a summary of the figures related to the macro analysis of the stemming algorithm.

Table 3. Macro comparison among runs.

	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

Note that 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. It is interesting to note that precision increases as well, and then the overall performance is higher thanks to the application of stemming than when it is without any stemmer. As previous studies on other non-English languages showed, [15,10] this interesting result suggests that stemming does not cause any trade-off between recall and precision, i.e. both precision and recall can be increased. To confirm the increase of effectiveness, Figure 2 shows the Averaged Recall-Precision curve at different levels of recall.

As regards 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 built without

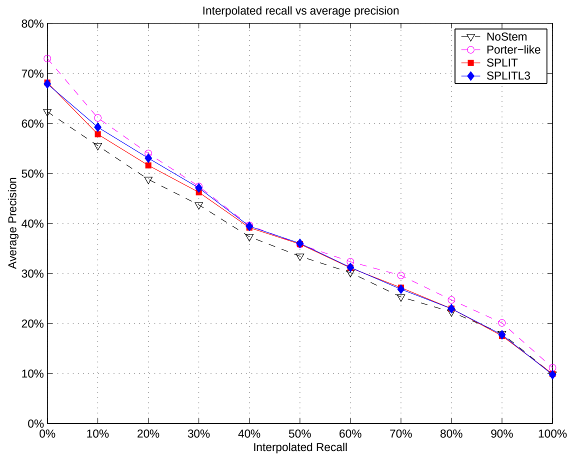


Fig. 2. Average Precision Curve for four stemming algorithms.

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.

After analyzing the results obtained at macro level, we realized that performance varied with query and that `SPLIT` performed better than `Porter-like` for a subset of queries. This variation led us to carry out the analysis reported in the next Section.

3.4 Query-by-Query Evaluation

We conducted a more specific analysis based on the evaluation of the stemming effects on each query of the test collection, by calculating the R-Precision and Average-Precision figures for each query and for each run. We carried out the analysis for `Porter-like` and `SPLIT-L3`; the latter was chosen because it performed a little better than `SPLIT`, yet the difference was not statistically significant. Table 4 reports the number of queries in which a stemming algorithm improved, decreased or kept as equivalent R-precision and Average Precision with respects to the "no-stemming" case.

As Table 4 shows, the number of queries showing improvements in performance after the stemming process is greater than the number of queries for which precision decreased. However, the improvement is not strong enough to be considered statistically significant. Moreover, all the stemming algorithms yield comparable results in terms of R-Precision and Average Precision, as the Wilcoxon test suggests for $\alpha = 0.05$. This means that the number of queries for which `Porter-like` performed better than `SPLIT` is comparable to, i.e. not statistically different from, the number of queries for which `SPLIT` performed better

Table 4. Behavior of the algorithms compared with non-stemming.

	R-Precision		Avg-Precision	
	SPLIT-L3	Porter-like	SPLIT-L3	Porter-like
Improved	19	17	26	26
Decreased	13	15	19	19
Equivalent	15	15	2	2

than Porter-like. In other words, SPLIT and Porter-like are equivalently effective.

The latter way of considering improvements corresponds to assess performance from a more user-oriented than system-oriented point of view. If stemming is applied in an interactive context, such as that of Digital Libraries applications, the ranking used to display the results to the user acquire a great importance – from a practical rather than theoretical point of view at least. In fact, it would more interesting to know if the end user finds the relevant document after 10 or 20 retrieved documents instead of knowing if successful retrieval is reached after 50% retrieved documents. To assess stemming performance from a more user-oriented point of view, we were interested in evaluating how the observed improvement of effectiveness thanks to stemming can change the ranking of retrieved documents. Hence, we compared precision at 10, 20, 30 document cutoff, as suggested in [8]. The paired Wilcoxon test suggests to reject the null hypothesis that stemming has no effect on performance; on the contrary, we can confirm the hypothesis that stemming improves the results. Table 5 reports the number of queries in which a stemming algorithm improved, decreased or kept as equivalent the Precision figure computed at 10, 20 and 30 retrieved relevant documents.

Table 5. Behavior of the algorithms compared with the baseline of non-stemming.

	SPLIT-L3			Porter-like		
	10 docs	20 docs	30 docs	10 docs	20 docs	30 docs
Improved	14	19	18	20	23	19
Decreased	7	12	14	8	9	13
Equivalent	26	16	15	19	19	15

The test gives the same results both for the SPLIT-L3 and the Porter-like algorithm at all document cutoff values. To confirm that a statistical and link-based stemming algorithm can be successfully used instead of a-priori linguistic knowledge, we compared the SPLIT-L3 with the Porter-like algorithm for the document cutoff values selected above. Then, we computed the p-value, which is a measure of the probability that the observed difference between the two algorithms could have occurred by chance. We noted that the performances of such algorithms are so close that the p-value of Wilcoxon test for 20 and 30 document cutoff values are over 90%. This means that it is almost certain that the observed difference between the two algorithms occurred by chance, i.e. that there is not any "structural" reason so that the two algorithms are different. Table 6 reports the effects on the queries of `SPLIT-L3` algorithm on `Porter-Like` baseline. For precision, in 10 relevant documents retrieved, the

Table 6. Behavior of `SPLIT-L3` on `Porter-like` baseline

	10 docs	20 docs	30 docs
Improved	8	16	13
Decreased	13	19	15
Equivalent	26	12	19

`Porter-like` algorithm performs better than `SPLIT-L3` algorithm, this means that `Porter-like` is more effective if very few documents are seen; if more than a few documents are seen, `SPLIT` performs similarly.

4 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 graph-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. In parallel, theoretical work will disclose properties that permit us to improve `SPLIT`. Finally, further experimental work is in progress with other languages.

Acknowledgments. The authors thank Carol Peters for the support and the colleagues of the Information Management Systems group of the Department of Information Engineering for the useful discussions and suggestions.

Massimo Melucci's work has been partially supported by the University of Padua under the 2000 research program for young researchers (*Progetti di ricerca per giovani ricercatori*).

References

1. M. Agosti, M. Bacchin, N. Ferro and M. Melucci. University of Padua at CLEF 2002: Experiments to evaluate a statistical stemming algorithm. In *Cross-Language Information Retrieval and Evaluation: Proceedings of the CLEF 2002 workshop*, Lecture Notes in Computer Science series, Springer Verlag (forthcoming).
2. 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.
3. CLEF Consortium. CLEF: Cross-Language Evaluation Forum. `http://www.clef-campaign.org`, 2002.
4. 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.
5. W.B. Frakes and R. Baeza-Yates. *Information Retrieval: data structures and algorithms*. Prentice Hall, 1992.
6. J. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):154–198, 2001.
7. M. Hafer and S. Weiss. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371–385, 1994.
8. D. Harman. How effective is suffixing? *Journal of the American Society for Information Science*, 42(1):7–15, 1991.
9. J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.
10. R. Krovetz. Viewing Morphology as an Inference Process,. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, 1993.
11. J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.
12. The Jakarta Project. Lucene. `http://jakarta.apache.org/lucene/docs/index.html`, 2002.
13. C.D. Manning and H. Schütze. *Foundations of statistical natural language processing*. The MIT Press, 1999.
14. C.D. Paice. Another Stemmer. In *ACM SIGIR Forum*, 24, 56–61, 1990.
15. 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.
16. C. Peters and M. Braschler. Cross-Language System Evaluation: the CLEF Campaigns. *Journal of the American Society for Information Science and Technology*, 52(12):1067–1072, 2001.
17. M. Porter. Snowball: A language for stemming algorithms. `http://snowball.sourceforge.net`, 2001.
18. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
19. G. Salton and M. McGill. *Introduction to modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.
20. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
21. Institut interfacultaire d'informatique. CLEF and Multilingual information retrieval. University of Neuchatel. `http://www.unine.ch/info/clef/`, 2002.
22. C. Buckley. Trec_eval. `ftp://ftp.cs.cornell.edu/pub/smart/`, 2002.