Decker and Lenz (Eds.): Advances in Data Analysis
Selected Papers from the 30th Annual Conference of the German Classification Society (GfKl)
Berlin, ISBN 978-3-540-70980-0, pp. 367-374, ©Springer 2007.

Putting Successor Variety Stemming to Work

Benno Stein and Martin Potthast

Faculty of Media, Media Systems
Bauhaus University Weimar, 99421 Weimar, Germany
{benno.stein | martin.potthast}@medien.uni-weimar.de

Abstract. Stemming algorithms find canonical forms for inflected words, e. g. for declined nouns or conjugated verbs. Since such a unification of words with respect to gender, number, time, and case is a language-specific issue, stemming algorithms operationalize a set of linguistically motivated rules for the language in question. The most well-known rule-based algorithm for the English language is from Porter [14].

The paper presents a statistical stemming approach which is based on the analysis of the distribution of word prefixes in a document collection, and which thus is widely language-independent. In particular, our approach addresses the problem of index construction for multi-lingual documents. Related work for statistical stemming focuses either on stemming quality [2,3] or on runtime performance [11], but neither provides a reasonable tradeoff between both. For selected retrieval tasks under vector-based document models we report on new results related to stemming quality and collection size dependency.

Interestingly, successor variety stemming has neither been investigated under similarity concerns for index construction nor is it applied as a technology in current retrieval applications. As our results will show, this disregard is not justified.

1 Introduction

Most of the words in a text document have various morphological variants. Since the variants have a similar semantics they can be considered as equivalent for the purpose of many retrieval tasks. Consider for example the words "connecting" and "connect": they are not recognized being equivalent without having them reduced to their stem. A *stem* is the portion of a word that is common to a set of inflected forms; it is not further analyzable into meaningful elements and carries the principle portion of meaning of the words in which it functions. *Stemming* is the process of reducing a word to its stem, and a *stemmer* or a stemming algorithm is a computer program that automates the task of stemming. As illustrated in Figure 1 stemming happens at an early stage in the text processing chain.

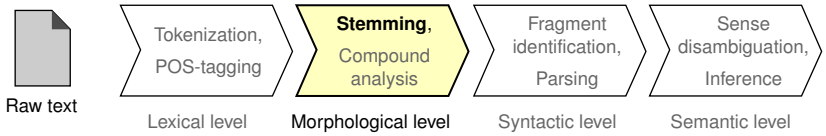

Fig. 1. The role of stemming in the text processing chain.

1.1 On Stemming

Since natural languages are irregular, stemming algorithms have a heuristic component and hence are subject to generating incorrect stems. One refers to overstemming, if too much of a word is removed, and to understemming, if words could be conflated to the same stem but may remain distinct after stemming. An example for overstemming is "relati-ve" / "relati-vistic", an example for understemming is "sensib-le" / "sensibili-ty". In the literature on the subject stemming algorithms are often judged by counting the number of stemming mistakes produced. However, we believe that it is reasonable to measure the power of a stemming approach by its impact on the performance of dedicated information retrieval tasks.

The different stemming strategies developed in the past can be classified according to the following scheme (cf. Figure 2):

- *Table Lookup.* Stores to each stem all flections in a hash table. Problems with this approach include the handling of non-standard words and storage requirements.
- *Truncate Stemming.* Retains the first k letters of a word, where k is a suitable integer; a word with less than k letters is simply returned.
- *Rule-based Affix Elimination.* Removes the match related to a precondition of a rule and possibly repairs the resulting stem. An important variant is iterative longest match stemming, which forms the base of several well-known stemming algorithms [14,13,10,9]. Note that these approaches are tailored to the English language; a recent development is the Snowball initiative, which employs language-specific rules [15].

Successor variety analysis is a special form of truncate stemming that applies a variable, say, word-specific k computed from the underlying collection. Compared to rule-based affix elimination a successor variety analysis is a purely syntactic approach and hence it should be inferior to a knowledge-based

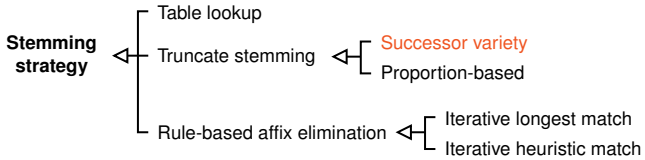

Fig. 2. Taxonomy of stemming strategies.

stemmer. At closer inspection the situation looks differently: (*i*) Successor variety analysis is adaptive and conservative by nature since it identifies and applies collection-specific stemming rules. (*ii*) A rule-based approach is problematic if, for instance, the document language is unknown, if no language-specific stemming rules are at hand, or if a document combines passages from several languages. Successor variety analysis is unaffected by this.

1.2 Contributions

Despite its advantages successor variety analysis is not applied as a technology for index refinement in current retrieval applications. In addition, only few and rather outdated analyses have considered this approach in their evaluations. The paper in hand addresses this gap. We have developed an implementation for successor variety stemming along with new pruning heuristics, which is compared to well-known rule-based stemming implementations. In particular, the following questions are investigated:

1. How far is successor variety stemming behind rule-based stemming for the languages English and German?
2. What is the quantitative connection between the quality of successor variety stemming and corpus size?[1]

To answer these questions we have set up a large number of text categorization experiments with different clustering algorithms. Since these algorithms are susceptible to various side effects, we will also present results that rely on an objective similarity assessment statistic: the measure of expected density, $\overline{\rho}$ [17].

The remainder of the paper is organized as follows. Section 2 introduces and discusses successor variety stemming, and Section 3 reports on selected classification experiments and similarity analyses.

2 Successor Variety Stemming

Given a set of a word's morphological variants, a potential stem can be derived heuristically, by a skillful analysis of prefix frequency and prefix length among the variants. E. g., the longest common prefix of the words "connection", "connect", "connectivity", "connecting" is "connect", which is also the stem. This principle applies to other languages like German as well: "verbinden", "Verbindung", "verbinde", "verbindend" all share the same stem "verbind". Of course there are exceptions in the German language which fall not into this scheme, such as the past principle "verbunden" in the example.[2]

[1] This question addresses also the issue of a "break-even point", above which one gets a pay-off from one or the other strategy.

[2] This idea can be extended to the identification of compound word concepts in a document. If continuous sequences of n words occur significantly often, then it

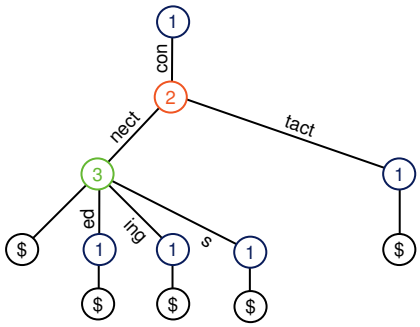

Fig. 3. A suffix tree at the character level for the words "connect", "connected", "connecting", "connects", and "contact". The inner nodes hold the frequency information, the $-sign denotes the end of string.

To identify significant prefixes we have developed an analysis method based on the suffix tree data structure [18,7], into which each word w of each document d of a collection D is inserted. The construction of a suffix tree is straightforward: A word w is inserted by testing whether some edge emanating from the root node is labeled with w's first character c_1. If so, this edge is traversed and the procedure is repeated by testing whether some edge of the successor node is labeled with c_2. If, at some depth k, a node n without a matching edge is reached a new node is created and connected to n with an edge labeled c_{k+1}. Frequency information is updated during suffix tree insertion. Figure 3 shows a suffix tree for morphological variants of the word "connect"; inner nodes with an outdegree of 1 are omitted. Note that suffix trees are used here because of their small memory footprint; with respect to runtime they may be outperformed by the Patricia data structure [12].

It remains to be answered how *reasonable* stems can be identified using a suffix tree. Obviously, a possible stem lies on a path that starts at the root, and, inner nodes that have a high outdegree—compared to their immediate predecessors—may be good stemming candidates. Strategies to operationalize this observation can be found in [6]:

- *Cutoff Method.* Take all subsequences of w as candidates that start at the root and end in a node with more successors than a given threshold. It remains unclear how the threshold has to be chosen; particularly it will depend on w's length.
- *Peak and Plateau Method.* Choose subsequences of w that end in a node that has more successors than its predecessor.

is likely that these words form a concept. Stemming and concept identification are essentially the same—the level of granularity makes the difference: Stemming means frequency analysis at the level of characters; likewise, the identification of concepts means frequency analysis at the level of words.

- *Complete Word Method.* Define as stem of a word w the shortest substring w' of w, which also occurs as a word in a document.
- *Entropy Method.* Choose subsequences w' of w whose entropy is significantly higher than the entropy of its immediate predecessor.

The strategy that has been applied in this paper is an enhancement of the peak and plateau method with respect to different subsequence lengths. To accept the substring of a word up to the k-th letter as suitable stem, the successor variety values v_{k-1}, v_k, and v_{k+1} must fulfill the following inequalities:

$$k > x \cdot m \quad \text{with} \quad x \in (0;1) \tag{1}$$

$$\frac{v_k}{v_{k-1}} > y \quad \text{with} \quad y > 0 \tag{2}$$

$$\frac{v_{k+1}}{v_k} < z \cdot \frac{v_k}{v_{k-1}} \quad \text{with} \quad z > 0 \tag{3}$$

The first inequality ensures a minimum length for a stem. The second inequality is a refinement of the peak and plateau method that simply claims the constraint $y = 1$. In the form presented here especially the case $0 < y < 1$, where the successor variety v_k is significantly high but not larger than v_{k-1}, can be recognized. The third inequality ensures that a word is not overstemmed given the case that a succeeding character provides a reasonable clipping position as well. Note that all computations within the analysis can be done during a single depth-first traversal of the suffix-tree, which leads to an overall time complexity comparable to Porter's algorithm.

Table 4 shows stems obtained by our successor variety analysis, the respective stems returned by Porter's algorithm, and the optimum stems.

Successor variety		**Porter**		**Optimum**
Stem(s)	Affix(es)	Stem(s)	Affix(es)	
minist	- ers, erial, er, - ry, ries	minist ministri	- ers, er - $, es	minist
oper operat operati	 - ors, or, es, e - onal, ons, ng, on	oper	- $, ators, ational, - ator, ates, ations, - ating, ation, ate	oper operat
exten	- sion, d, ds, ding, - t, ded, sive	extend extens extent	- $, s, ing, ed - ion, ive	exten

Fig. 4. The table contrasts stems obtained by our successor variety analysis and by Porter's algorithm with optimum stems. The $-symbol denotes the empty string.

3 Quantitative Analysis of Stemming Approaches

Only few experiments related to stemming were made in the past; in particular, existing research investigates neither the quality of successor variety stemming nor its language independence[3] [5,8,6,4,1]. In this connection the purpose of our stemming analyses is twofold. (*i*) We want to analyze the potential of successor variety stemming compared to rule-based stemming. (*ii*) Since successor variety stemming is expected to be collection-size-dependent by nature, the trade-off between stemming quality and collection size shall be revealed. The employed document model is the vector space model; stopwords are omitted and term weighting is done according to the $tf \cdot idf$-scheme.

The analyses illustrate the impact of a stemming strategy in two ways: *Indirectly*, by comparing the classification performance within a categorization task, expressed by the F-Measure, and *directly*, by comparing the intrinsic similarity relations captured within a document model, expressed by $\overline{\rho}$: Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be an exclusive categorization of D, and let $G = \langle V, E, \varphi \rangle$ be the underlying similarity graph of D. Based on the global edge density θ of G, $\overline{\rho}$ averages the class-specific density improvement within the subgraphs $G_i = \langle V_i, E_i, \varphi \rangle$ induced by the categories C_i:

$$\overline{\rho}(\mathcal{C}) = \sum_{i=1}^{k} \frac{|V_i|}{|V|} \cdot \frac{\sum_{e \in E_i} \varphi(e)}{|V_i|^{\theta}}, \quad \text{where } \theta \text{ computes from } |V|^{\theta} = \sum_{e \in E} \varphi(e)$$

If for a collection of documents the $\overline{\rho}$-value under document model M_2 is larger than the $\overline{\rho}$-value under document model M_1, then M_2 captures more of the intrinsic similarity structure of the collection [17].

The experiments rely on RCV1, a short hand for "Reuters Corpus Volume 1" [16], as well as on a corpus of German newsgroup postings. RCV1 was published by the Reuters Corporation for research purposes; it contains about 800,000 documents each of which consisting of a few hundred up to several thousands words. The documents are tagged with meta information like category (also called topic), geographic region, or industry sector. The German newsgroup corpus has been compiled in our working group and comprises 26,000 postings taken from 20 different newsgroups. From these corpora the samples were formed as follows: For the analysis of the categorization tasks the sample sizes were 1000 documents, chosen from 10 categories. For the analysis of the intrinsic similarity relations the sample sizes were ranging from 200 to 1000 documents, chosen from 5 categories.

Figure 5 shows selected analysis results for the two languages English (left-hand side) and German (right-hand side). The tables comprise the effects of the different document models within the categorization task, expressed as improvements in the F-Measure-value for different cluster algorithms. The

[3] Language independence, however, applies primarily to languages whose flections base on suffixes, prefixes or circumfixes.

Stemming strategy	F-min (sample size 1000, 10 categories)	F-max	F-av.
without	—baseline—		
Porter	-11%	14%	6%
successor variety	-15%	14%	4%

Stemming strategy	F-min (sample size 1000, 10 categories)	F-max	F-av.
without	—baseline—		
Snowball	-8%	26%	10%
successor variety	-9%	23%	6%

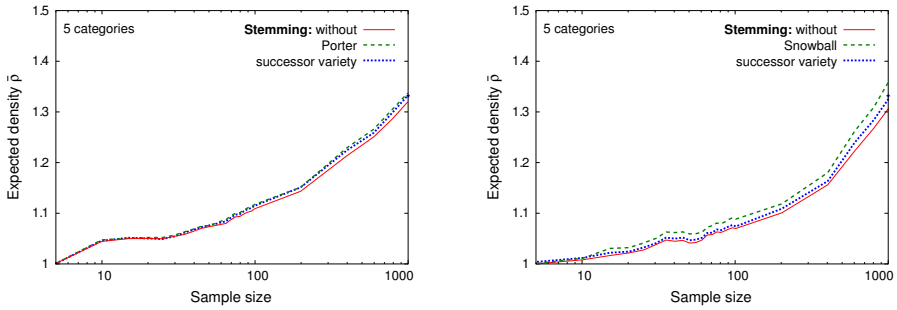

Fig. 5. The effect of no stemming, rule-based stemming, and successor variety stemming on the vector space model, given an English collection (left) and a German collection (right). The tables (top) comprise the effects within the categorization task; the graphs below show the relation between the collection size and the similarity relations captured within the document model.

graphs show that—independent of the stemming strategy—the collection size is positively correlated with the expected density $\overline{\rho}$. This means that additional documents add similarity information rather than noise. Altogether, rule-based stemming performs slightly better than successor variety stemming, which shows a reasonable performance though.[4]

4 Discussion

This paper reported on a comparison between rule-based stemming and successor variety stemming by evaluating the retrieval performance of the respective vector space models. Our approach to successor variety stemming is based on a suffix tree data structure, and controlled by new pruning heuristics that skillfully analyze the successor variety of the inner tree nodes.

For the performance evaluation both an indirect method and a direct method has been applied. The former relies on the application of clustering algorithms in a text categorization task; the latter relies on the similarity graph of a document collection and quantifies improvements between the inter-class and the intra-class variance of the edge weights. The following analysis results shall be emphasized:

[4] To make our analysis results reproducible for other researchers, meta information files that describe the compiled test collections have been recorded; they are available upon request.

1. The effect of stemming with respect to the vector space model is less than commonly expected.
2. Compared to rule-based stemming, the retrieval performance of our optimized successor variety stemming is only slightly worse. Note that for the German language this performance can be further improved by applying the same strategy to identify prefixes and by adjusting the pruning heuristics to identify compound words.

Two salient advantages of successor variety stemming are its language independence, and its robustness with respect to multi-lingual documents. An obvious disadvantage may be the necessary statistical mass: Successor variety stemming cannot work if only few, very small document snippets are involved. This effect could directly be observed in our experiments.

References

1. S. Abdou, P. Ruch, and J. Savoy. Evaluation of stemming, query expansion and manual indexing approaches for the genomic task. In *Proc. TREC'05*, USA, 2005.
2. M. Bacchin, N. Ferro, and M. Melucci. Experiments to Evaluate a Statistical Stemming Algorithm. `http://clef.isti.cnr.it/2002work.html`, 2002.
3. S. Bordag. Unsupervised knowledge-free morpheme boundary detection. In *Proc. of the RANLP'05*, 2005.
4. M. Braschler and B. Ripplinger. How effective is stemming and decompounding for german text retrieval? *Information Retrieval*, 7(3-4):291–316, 2004.
5. W. B. Frakes. Term conflation for information retrieval. In *Proc. of SIGIR'84*, pages 383–389, Swinton, UK, 1984. British Computer Society.
6. W. B. Frakes and R. Baeza-Yates. *Information retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.
7. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
8. D. Harman. How effective is suffixing? *J. of the ASIS&T.*, 42(1):7–15, 1991.
9. R. Krovetz. Viewing morphology as an inference process. In *Proc. of SIGIR'93*, pages 191–202, New York, NY, USA, 1993. ACM Press.
10. J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computation Linguistics*, 11(1):23–31, March 1968.
11. J. Mayfield and P. McNamee. Single n-gram stemming. In *Proc. of SIGIR'03*, pages 415–416, New York, NY, USA, 2003. ACM Press.
12. D. R. Morrison. PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric. *J. of the ACM*, 15(4):514–534, October 1968.
13. C. D. Paice. Another stemmer. *SIGIR Forum*, 24(3):56–61, 1990.
14. M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.
15. M. Porter. Snowball. `http://snowball.tartarus.org/`, 2001.
16. T.G. Rose, M. Stevenson, M. Whitehead. The Reuters Corpus Volume 1 - From Yesterday's News to Tomorrow's Language Resources. In *Proc. LREC'02*, 2002.
17. B. Stein, S. Meyer zu Eißen, and F. Wißbrock. On Cluster Validity and the Information Need of Users. In *Proc. of AIA'03*, pages 216–221, September 2003.
18. P. Weiner. Linear pattern matching algorithm. *Proc. of the 14th IEEE Symp. on Switching and Automata Theory*, pages 1–11, 1973.