Analysis and Algorithms for Stemming Inversion

Ingo Feinerer

Vienna University of Technology, Austria
ingo.feinerer@tuwien.ac.at

Abstract. Stemming is a fundamental technique for processing large amounts of data in information retrieval and text mining. However, after processing the reversal of this process is often desirable, e.g., for human interpretation, or methods which operate on sequences of characters. We present a formal analysis of the stemming inversion problem, and show that the underlying optimization problem capturing conceptual groups as known from under- and overstemming, is of high computational complexity. We present efficient heuristic algorithms for practical application in information retrieval and test our approach on real data.

Keywords: Stemming, inversion.

1 Introduction

Stemming has been widely used as a fundamental technique in information retrieval, especially in the context of web search engines [19], in text mining [20] and sentiment analysis [1] as a preprocessing technique, and for databases when building large indices on documents [18]. The main reasons are reduced memory and computing demands which are necessary to handle large amounts of data, however once stemmed it is hard for humans to work with the underlying material for manual inspection. Further, methods which directly operate on character sequences, like string kernels [10], or computer-assisted approaches for content analysis of textual data which involve exact word matching, like the General Inquirer [17], may return unexpected results. One can address this problem in multiple ways. First, we could avoid the "design error" of deleting information we need later on, e.g., store both the original and stemmed representation. However, this introduces significant space overhead for large corpora, and ignores the fact that many text mining routines only return stems due to their internal representation. Second, one can use the context, i.e. the order of the stems appearing in the original texts, to better reconstruct the original words. Computational morphology also provides techniques for implementing reversible methods, nevertheless, the stemming process cannot be directly reverted. A far more conservative assumption, which will ground our further considerations, is that the original words to stems can only be restored from a dictionary. This is especially relevant for large amounts of (very) short texts (like blogs or Twitter) as context is hardly preserved due to abbreviations or stemming. Based on a dictionary the main challenge is now to find completions for individual stems along compatible semantic groups.

P.-J. Cheng et al. (Eds.): AIRS 2010, LNCS 6458, pp. 290–299, 2010.

2 Stemming

Stemming denotes the process of conflating words to their stems, e.g. by deleting word suffixes. Formally stemming can be seen as a surjective function $s\colon \Sigma^* \to \Sigma^*$ mapping words from an alphabet Σ to words from the same alphabet. Stemming functions can be evaluated e.g. by under- and overstemming errors [14], by checking whether words of the same conceptual group are actually conflated to the same stem.

Example 1. Table 1 shows words (completions) and their word stems (c_i and s_i are shortcuts for completions and stems, respectively), and the effect of over- vs. understemming. All words starting with `exp` are conflated to the same stem `experi` ($= s_1$) although the first two belong to another conceptual group (*overstemming*) as the rest, whereas `adhe` words have different stems (s_2 and s_3) although describing a similar semantic concept (*understemming*).

Table 1. Completions and stems

C	Completion Word	Stem	S
c_1	experimental		
c_2	experiment	experi	s_1
c_3	experience		
c_4	experiences		
c_5	adhere	adher	s_2
c_6	adhesion	adhes	s_3

Prominent stemmer implementations are the Porter [15], Lovins [11], Paice [13], Dawson [2] and Krovetz [7] stemming algorithms.

3 The Stemming Inversion Problem

Given two sets of terms—word stems and possible completions—we want to find an assignment of stems to completions in such a way that as many of the original, i.e. before initial stemming, words (or at least its semantics) are restored. Conceptually, the best solution to this optimization problem is exactly the inversion of the implemented stemming procedure. However, since stemming is surjective a one-to-one inversion is in general not always possible, and we need a notion of optimality in terms of choosing completions for given stems. We formalize this as follows:

Definition 1 (Stemming Inversion Problem). *Let $\mathcal{S}$ be a set of stems, and $\mathcal{C}$ be a set of completions. Note that both sets are not necessarily disjoint, i.e. $\mathcal{S} \cap \mathcal{C} \neq \emptyset$. Let $\mathcal{G}_\mathcal{C}$ be a collection of sets (groups) of completions of related semantic interpretation, and there may exist sets of stems $\mathcal{G}_\mathcal{S}$ which group stems of related meaning (e.g., when stems are considered as equivalent which is useful for combining stemming and lemmatization tasks). The sets in $\mathcal{G}_\mathcal{C}$ can be used to*

indicate that a single completion of the same group suffices to participate in the matching. Finally, let $\mathcal{L}$ *contain all* links *between stems and valid corresponding possible completions. Note that not all stems need to have a link to a completion (e.g., if there is a completion missing due to an incomplete dictionary).*

The stemming inversion problem *is to choose the minimal number of links* $l \in \mathcal{L}$ *and groups in* $\mathcal{G}_\mathcal{C}$, $\mathcal{G}_\mathcal{S}$, *such that all stems* $s \in \mathcal{S}$ *and completions* $c \in \mathcal{C}$ *which are present in at least one link or group in* $\mathcal{G}_\mathcal{C}$, $\mathcal{G}_\mathcal{S}$ *in the input, are covered. An element is covered if it is present in at least one link or group in the solution.*

Note that this definition allows isolated stems or completions (and e.g. $\mathcal{C}$ to be a dictionary) which are simply ignored. Also note that it is allowed to choose multiple completions for a given stem. We present completion strategies for choosing a single element out of multiple completions at the end of this section.

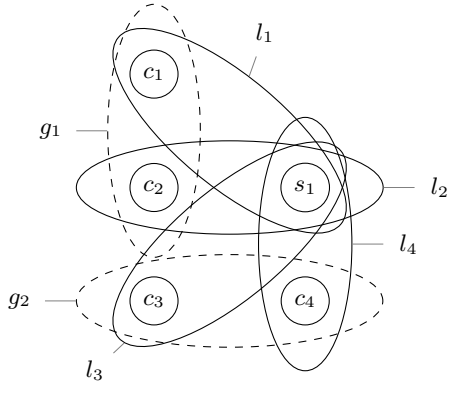

Fig. 1. A subset of the inversion problem instance of Ex. 2 for the `experi*` cluster

Example 2. A corresponding instance to the stemming inversion problem for Ex. 1 is defined by $\mathcal{S} = \{s_1, s_2, s_3\}$, $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, $\mathcal{G}_\mathcal{C} = \{g_1 = \{c_1, c_2\}, g_2 = \{c_3, c_4\}, g_3 = \{c_5, c_6\}\}$, $\mathcal{G}_\mathcal{S} = \emptyset$, and $\mathcal{L} = \{l_1 = \{c_1, s_1\}, l_2 = \{c_2, s_1\}, l_3 = \{c_3, s_1\}, l_4 = \{c_4, s_1\}, l_5 = \{c_5, s_2\}, l_6 = \{c_6, s_3\}\}$, where the c_i and s_i correspond to the shortcuts defined in Tab. 1. Figure 1 depicts the situation for the subset induced by the terms starting with `experi`. The assignment of stems, completions, and links (solid lines) between stems and possible completions follows directly by definition. The assignment of groups (dashed lines) is given by the observation that c_1 and c_2 have a similar semantics as a completion, whereas c_3 and c_4 are different and thus are in another group. Similarly, we define c_5 and c_6 to be in the same semantic group. A solution to this instance is given by choosing the links l_3, l_5, l_6, and the groups g_1 and g_2, as this covers all stems and completions. Note that the groups allow us to choose one out of several choices, but only if the links allow us this.

This formalism allows us to model important notions for stemming inversion. However this comes at a computationally high price as we show the problem to be NP-hard via a polynomial-time reduction from the set cover problem.

Definition 2 (Set cover problem). *Let U be a universe of elements, and S be a family of subsets of U. The* set cover optimization problem *is now to find the minimal number of subsets T_i of S such that $\bigcup T_i = U$.*

This problem is NP-hard [6,4]. For the reduction we need to encode every possible instance of the set cover problem to our stemming inversion problem:

Proof. For a given set cover problem with the universe U we define $\mathcal{C} = U$, i.e., all elements of the set cover problem are assumed to be completions, and we set $\mathcal{G}_\mathcal{C} = S$, i.e., the groups of the completions directly correspond to the subsets of the set cover problem. □

Note that we do not even need links between completions and stems since already the problem of choosing a minimal number of semantically equivalent groups is expensive. However we argue that the inversion problem in its full generality cannot be simplified as we observe that stemming functions can be arbitrary surjective functions, and as such can have one or multiple completions, and that groups of completions and stems depend on semantic notions and language properties, and as such can overlap and intersect each other.

Nevertheless, in many languages, the underlying problem is not that complex since there is normally no deep nesting or overlapping between groups. That means that naive brute-force algorithms will work reasonably well in practical applications since the search space that needs to be explored is separated in local components (e.g. see the two clusters in Ex. 1).

So far we were only concerned about finding optimal assignments between stems and completions respecting the semantics of related conceptual groups. In addition we need a local criterion which tells us which element should be chosen out of multiple completions. E.g., in Ex. 2 we could prefer to choose c_3 or c_4 out of group g_2 as completion for stem s_1, depending on our notions of optimality. A description of possible completion heuristics follows:

Prevalent. Choose the completion c with the maximal number of occurrences in a corpus, i.e. such that $\max_{i=1,\ldots,n}|c_i| = c$, where n denotes the total number of completions for a single stem.

First. Choose the completion c which is first found in a dictionary, i.e., $c_1 = c$.

Shortest. Choose the completion c with the minimal number of characters of all suitable completions, i.e., $\min_{i=1,\ldots,n}|\mathrm{char}(c_i)| = c$, where n denotes the total number of completions for a given term.

Longest. Symmetric to the previous case, i.e., the completion with the maximal number of characters: $\max_{i=1,\ldots,n}|\mathrm{char}(c_i)| = c$.

Random. Choose a random item c out of the possible completions, i.e., $c_i = c$ with $i \in \{1, \ldots, n\}$.

The actual implementation of the strategies is slightly more complicated due to some intricacies of prominent stemming algorithms. In detail it does not suffice just to search for completions only but also consider insertion or deletion of characters. E.g., the Porter stemming algorithms produces `berri` out of `berry`. A solution is to use completions with minimal Levenshtein distance [8] regarding insertion/deletion for such special cases.

4 Experiment

To evaluate our presented approaches we implement a benchmark consisting of three test suites working on real data in an information retrieval setting.

Reconstruction. Given a text corpus we build a dictionary consisting of the terms. Successively, we draw samples of the terms in the corpus, and apply the Porter [15] stemming algorithm. The stems are now completed with the different inversion heuristics presented in the previous section. The results from the completion are compared to the original samples, and a matching statistics is computed. This gives us information on how much information from the original corpus can directly be reconstructed.

Text clustering. Our second test suite measures the performance of our approximation heuristics if applied to text clustering. We cluster the original, the stemmed, and each corpus obtained by applying every completion heuristic on top of the stemmed version. In detail we use the classical k-means algorithm [5,12] which minimizes the objective function $\sum_{j=1}^{k} \sum_{x_i \in \pi_j} \|x_i - m_j\|^2$, where π_j are clusters and $m_j = \frac{1}{\|\pi_j\|} \sum_{x_i \in \pi_j} x_i$ is the mean of cluster π_j.

Sentiment Analysis. Our final procedure tests for the effect of inversion in a slightly more involved scenario. We implement a sentiment analysis using word lists as found in the General Inquirer [17], both on the stemmed and the completed words in our data set, and compare the results. Sentiment analysis can be seen as text mining for opinions, and is typically implemented by counting words of specific categories (e.g., the General Inquirer word list for the *Positive* category includes "good", "extraordinary", or "outstanding"), and computing scores (e.g., the sum) for each category $\mathcal{C}_j$: $\text{score}(\mathcal{C}_j, \tau) = \sum_{w_i \in \mathcal{C}_j} \Phi(w_i, \tau)$, where $\Phi(w, t)$ counts the frequency of a word w in a text t, and τ denotes the underlying corpus.

4.1 Data

We use the Reuters-21578 data set [9] as the basis for our experiments. It contains stories covering a broad range of topics, like mergers and acquisitions, finance, or politics, and was collected by the Reuters news agency. The data set is publicly available[1] and has been widely used in text mining and information retrieval within the last decades. It contains 21578 short to medium length documents.

4.2 Procedure

The experiment is carried out using the R [16] statistical computing and graphics environment since it provides a text mining infrastructure [3] which makes it easy to implement a prototype and apply user-defined algorithms, in our case the five presented completion heuristics, provides a broad spectrum of clustering and classification tools, and has support to access the General Inquirer word lists

[1] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

and score documents accordingly. For the first part (*Reconstruction*) we built a dictionary out of the corpus, resulting in—after some preprocessing like removal of punctuation marks, numbers, and common stopwords—about 45000 unique terms. Next, we drew samples consisting of 100 randomly chosen terms out of this dictionary. The terms were stemmed and heuristically completed with the five presented procedures. We independently repeated the sample, stemming, and completion steps up to 100 times, resulting in up to 100 different runs. The results of the individual runs are analyzed such that for a given sample we compare the completions of each individual completion heuristics with the original unstemmed terms, and compute the percentage of matching terms. This corresponds to the relative amount of terms where a perfect inversion (reconstruction) was possible. In the second experiment (*Text clustering*) we extracted all documents of the Reuters-21578 corpus with topics *acq* (acquisitions; 2125 documents) or *crude* (crude oil; 355 documents). These documents are the input for a k-means clustering procedure (averaging over 100 runs to compensate for local minima due to unfortunate (randomized) initial start assignments) with two desired clusters ($k = 2$, modeling the two chosen topics). Since we have the true class/topic ids from the annotations in the corpus, we compute cross-agreements using maximal co-classification rate (i.e., the maximal rate of objects with the same class ids in both clusterings). The third part (*Sentiment analysis*) implements a sentiment analysis by scoring individual documents according to occurrences of matching terms from word lists as provided by the General Inquirer. These word lists are carefully chosen for specific categories (we use *Positive*, *Negative*, *Strong*, *Weak*, *Active*, *Passive*, *Rise* and *Fall*) by the creators of the General Inquirer, and are as such highly sensitive to individual word forms. I.e., it might happen that a word matches, but its stem does not, but also vice versa. We then compute word scores for documents consisting of stemmed words and compare their scores when using our presented inversion heuristics.

5 Results

For the first part of our experiment, Fig. 2 depicts the percentage of matching terms for the five completion strategies `prevalent`, `first`, `shortest`, `longest`, and `random` for 15 independent runs. Note that although the runs are independent we connect points of the same method to visualize their relative positions among each other throughout the experiment. Also be aware that we only visualize 15 runs instead of 100 to see more details but the results hold for the full experiment in the same way. For comparison we also have a `none` entry which corresponds to the stemmed terms. I.e., if there is a match for `none` this means the stem and the completion is identical, which is the case for about 43% of the terms in average in Fig. 2. We see that `prevalent`, `first`, and `shortest` are comparable in performance with about 67% which corresponds to a relative increase of about 55%. Interestingly, even `random` outperforms `longest` which has the lowest matching rate of all strategies.

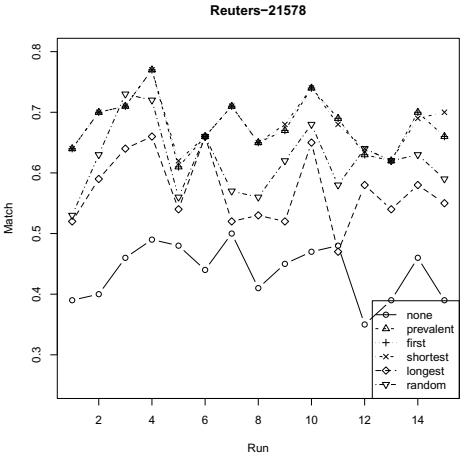

Fig. 2. Match results of the five stemming inversion heuristics for 15 independent runs

Table 2. Percentage of matches between original terms and stemmed terms with applied completion heuristic (*Reconstruction*), and percentage between cross-agreements using maximal co-classification rate for k-means ($k = 2$) clustering for stemmed and completed terms in the Reuters-21578 corpus (*Clustering*)

	Reconstruction	Clustering
none	43.95	0.00
prevalent	67.09	**15.99**
first	67.09	6.42
shortest	**67.46**	3.14
longest	56.60	-0.04
random	61.24	6.39

Table 2 gives exact numbers for this observed behavior. The column "Reconstruction" lists the percentage of matches between the original terms and the stemmed terms with completion for all 100 runs.

Column "Clustering" is relevant for our text clustering experiment and shows the relative performance of k-means clustering for the two ($k = 2$) clusters of known topics *acq* and *crude* both for the stemmed corpus and the corpora obtained by first applying our completion heuristics. Relative performance measures the cross-agreements (between cluster labels obtained from the k-means clustering procedure and the true class ids available as annotations from the corpora) using maximal co-classification rates. The zero entry for *none* forms the base line as no relative improvement can be achieved (obviously, since no heuristics is applied). Interestingly, *longest* has a negative impact on the clustering results. I.e., it is better to use no heuristics than this one on the Reuters-21578 data set. Strategies *shortest* and *first* perform about 3% and 6% better than

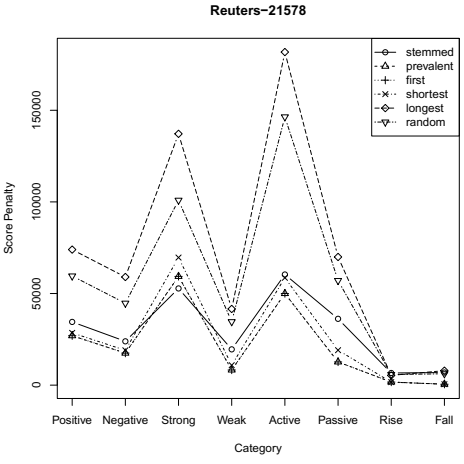

Fig. 3. Score penalties for the inversion heuristics for the categories *Positive*, *Negative*, *Strong*, *Weak*, *Active*, *Passive*, *Rise* and *Fall*

Table 3. Score penalties for the inversion heuristics for the categories *Positive*, *Negative*, *Strong*, *Weak*, *Active*, *Passive*, *Rise* and *Fall*

	prevalent	first	shortest	longest	random
Positive	**26,966**	**26,966**	28,513	73,971	59,567
Negative	**17,469**	**17,469**	19,017	59,020	44,716
Strong	**59,343**	**59,343**	69,626	137,152	100,899
Weak	**8,269**	**8,269**	10,888	41,577	34,617
Active	**49,995**	**49,995**	58,579	181,767	146,397
Passive	**12,658**	**12,658**	19,078	69,942	57,072
Rise	1,686	1,686	**1,629**	5,335	5,785
Fall	**505**	**505**	**505**	7,897	6,206

none. Heuristic *prevalent* outperforms all others with a relative performance increase of almost 16%.

For the third part, the application of our methods in the context of sentiment analysis, Fig. 3 depicts summarized score penalties for the whole Reuters-21578 corpus for individual categories from the General Inquirer word lists. Summarized score penalties $|\text{score}(\mathcal{C}_i, \tau_O) - \text{score}(\mathcal{C}_i, \tau_S)|$ for a category $\mathcal{C}_i$ correspond to the sum of word frequencies which are different from the original corpus τ_O considering that the stemmed corpus τ_S has negative impact on the individual scores. A penalty of 0 means that an inversion method produces exactly the same score as the original document, i.e., the smaller the penalty, the better. Table 3 shows corresponding numbers for Fig. 3.

By combining the results from the different conducted experiments we observe that the method `prevalent` performs best and yields a notable improvement

compared to just using a stemmed corpus, i.e., without any completion strategy. This result is in so far surprising since it holds in *all* of our experiments; it could be expected for reconstruction since the terms which occur most often are also likely to be the correct ones for completion. However, this is not necessarily true for text clustering, and especially not for text mining on sentiments. Clusterings are selective to large numbers of identical observations; a strategy in favor of prevalent terms can merge different stems to a single one occurring quite often. Similarly, joining different stems to a common word might trigger different semantics related to sentiments in text mining. Nevertheless, our experiments show that a prevalent strategy works in these two settings satisfactorily. For special settings `shortest` is a viable alternative, especially for runtime sensitive applications. Given a sorted dictionary finding the shortest completion is easy compared to determining the most prevalent terms in a corpus which involves counting word frequencies. We note that both the `random` (as expected) but also the `longest` completion strategy are unsatisfactory, and should not be used in general.

6 Conclusion

We presented the stemming inversion problem which is motivated by the fact that stemming is a central technique in information retrieval but the reversal of this process is often desirable, either for human consumption, or for word sensitive methods. Although being a natural and important research question, to the best of our knowledge this problem has not been addressed in the literature in a systematic way we do. We proved that the problem in its full generality is NP-hard. We introduced efficient approximation algorithms which solve the inversion problem based on a set of heuristics. Experiments on real data show the applicability of the presented heuristics for word reconstruction, text clustering, and text mining. We obtain strong results for a strategy based on prevalent term completions, yielding observable better results than just the stemmed words without completion heuristics, in all categories of our benchmarks. It remains open to investigate to which extent alternative strategies depend on the corpus language. This is highly relevant in an international context, e.g. for Internet search engines [19] due to user provided queries. Another remaining issue is the extension of our implementation into a stand-alone library. This would allow easy integration of the proposed inversion strategies into existing information retrieval frameworks. Finally there are plans to incorporate context-aware techniques in our setting.

Acknowledgments. This work is supported by the Vienna Science and Technology Fund (WWTF), project ICT08-032.

References

1. Annett, M., Kondrak, G.: A comparison of sentiment analysis techniques: Polarizing movie blogs. In: Bergler, S. (ed.) Canadian AI. LNCS (LNAI), vol. 5032, pp. 25–35. Springer, Heidelberg (2008)

2. Dawson, J.L.: Suffix removal for word conflation. Bulletin of the Association for Literary and Linguistic Computing 2(3), 33–46 (1974)
3. Feinerer, I., Hornik, K., Meyer, D.: Text mining infrastructure in R. Journal of Statistical Software 25(5), 1–54 (2008), http://www.jstatsoft.org/v25/i05
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
5. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: A K-means clustering algorithm (AS R39: 81V30 p355-356). Applied Statistics 28, 100–108 (1979)
6. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103 (1972)
7. Krovetz, R.: Viewing morphology as an inference process. Artificial Intelligence 118(1–2), 277–294 (2000)
8. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
9. Lewis, D.: Reuters-21578 text categorization test collection (1997), http://www.daviddlewis.com/resources/testcollections/reuters21578/
10. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. of Machine Learning Research 2, 419–444 (2002)
11. Lovins, J.B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics 11, 22–31 (1968)
12. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
13. Paice, C.D.: Another stemmer. SIGIR Forum 24(3), 56–61 (1990)
14. Paice, C.D.: Method for evaluation of stemming algorithms based on error counting. Journal of the American Society for Information Science 47(8), 632–649 (1996)
15. Porter, M.: An algorithm for suffix stripping. Program 3, 130–137 (1980)
16. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2010), http://www.R-project.org ISBN 3-900051-07-0
17. Stone, P.J.: Thematic text analysis: new agendas for analyzing text content. In: Text Analysis for the Social Sciences. ch. 2, Lawrence Erlbaum Associates, Mahwah (1997)
18. Strzalkowski, T., Vauthey, B.: Information retrieval using robust natural language processing. In: Proc. of the 30th annual meeting on ACL, Association for Computational Linguistics, Morristown, NJ, USA, pp. 104–111 (1992)
19. Uyar, A.: Google stemming mechanisms. J. of Inf. Sci. 35(5), 499–514 (2009)
20. Weiss, S., Indurkhya, N., Zhang, T., Damerau, F.: Text Mining: Predictive Methods for Analyzing Unstructured Information. Springer, Heidelberg (2004)