Distribution Based Stemmer Refinement

B.L. Narayan and Sankar K. Pal

Machine Intelligence Unit, Indian Statistical Institute,
203, B. T. Road, Calcutta - 700108, India
{bln_r, sankar}@isical.ac.in

Abstract. Stemming is a common preprocessing task applied to text corpora. Errors in this process may be refined either manually or based on a corpus. We describe a novel corpus-based stemming technique which models the given words as being generated from a multinomial distribution over the topics available in the corpus. A sequential hypothesis testing like procedure helps us group together distributionally similar words. This stemmer refines any given stemmer and its strength can be controlled with the help of two thresholds. A refinement based on the 20 Newsgroups data set shows that the proposed method splits equivalence classes appropriately.

1 Introduction

Stemming is the process of clubbing together words that are similar in nature. This process improves recall and reduces the dictionary size of a corpus. Several standard techniques are available in the literature which perform stemming [1]. The strength of a stemmer is the amount of reduction in the size of the dictionary obtained by it [1]. Strong (or aggressive) stemmers may reduce the size of a given corpus drastically, but may result in a severe decrease in precision. Stemming is afflicted with two kinds of errors: under-stemming and over-stemming. These errors are either refined manually or automatically with the help of a corpus.

In this article, we describe the design of a corpus-based stemmer which makes use of the class information of the corpus. We model words as arising from a multinomial distribution [2] and club distributionally similar words together. The equivalence classes thus generated represent the proposed stemmer.

The article is organized as follows. The background on stemming and related work is provided in Section 2. Section 3 describes the errors accompanying stemming and methods for refinement. Then, we describe the proposed stemming technique and present the experimental results in Sections 4 and 5, respectively. Conclusions and future work are discussed in Section 6.

2 Stemming and Related Work

Documents are generally represented in terms of the words they contain, as in the vector space model [3]. Many of these words are similar to each other in the sense that they denote the same concept(s), *i.e.*, they are semantically similar.

S.K. Pal et al. (Eds.): PReMI 2005, LNCS 3776, pp. 672–677, 2005.

Generally, morphologically similar words have similar semantic interpretations and may be considered as equivalent. The construction of such equivalence classes is known as stemming.

A number of stemming algorithms, or stemmers, have been developed, which attempt to reduce a word to its stem or root form. Thus, the document may now be represented by the stems rather than by the original words. As the variants of a term are now conflated to a single representative form, it also reduces the dictionary size, which is the number of distinct terms needed for representing a set of documents. A smaller dictionary size results in a saving of storage space and processing time.

Stemming is frequently used in the field of information retrieval [4], because it results in an increase in recall, as documents not containing the exact query terms are also retrieved. Moreover, the storage space for the corpus and retrieval times are reduced, without much loss of precision. Also, this give a system the option of query expansion to help a user refine his query.

Stemming also reduces the size of the feature set (when words are viewed as the features of documents). For the purpose of classification, this means that the models involved are far less complex than what would have been if the original set of words were used. This also means that it would lead to better generalization [5], in the sense that a small training error would imply a small test error too. It has been observed that the classification performance does not go down much due to the application of some of the standard stemmers. Also, this would lead to a reduction in the size of the corpus that needs to be stored.

Various stemmers are available in the literature. The most commonly used are the ones by Porter [6], Krovetz [7] and Paice/Husk [8]. Variants of these stemmers are available for non-English languages, too. The above stemmers apply a series of rules to a given word to obtain its stem.

3 Stemming Errors and Refinement

English, just like most other languages, is very diverse, and the morphological variants of all words cannot be obtained in the same manner. This leads to two kinds of stemming errors: under-stemming and over-stemming [9]. Under-stemming is the case where words that should have been grouped into the same class are not so. The more serious error is over-stemming, which results when too many unrelated words are merged together. This leads to a reduction in precision during retrieval and an increase in the error rate for classification. some examples of over-stemming by the Porter algorithm are:

- range and rang, even though unrelated, both stem to *rang*.
- petition, petit and petite are all stemmed to *petit* though petition is quite distinct from the rest.

These examples demonstrate how certain unrelated words may be grouped together on the basis of their morphological similarity. This problem is more pronounced in the case of stronger stemmers like Truncate(3) which clubs together all words whose first three letters are the same.

These errors may be manually corrected by specifying an exception list, or by means of modifying the rules of the stemmer. However, such refinements are labor intensive and may still not be appropriate. Moreover, the meanings and senses attached to a word vary a lot from corpus to corpus. An equivalence class considered appropriate for one corpus may not be so for a different one.

Xu and Croft devised a technique to automatically refine a stemmer. It was assumed that words with similar senses (which should be stemmed to the same stem) co-occur more often than dissimilar words. This co-occurrence information, obtained from a given corpus, is employed to split equivalence classes so that the resultant classes have larger co-occurrences between constituent words. The optimal algorithm that they provided for this purpose is computationally intensive and so, they opt for a suboptimal method based on connected component labeling to split large equivalence classes.

However, the basic assumption behind this co-occurrence based refinement, that similar words often co-occur may not hold in several instances. For example, documents describing an event of the past may use words related to the past tense only. Similarly, substitute words and word variants may not co-occur because an author of a document would generally stick to a single writing style throughout that document. The British and American variants of words form a prominent example of this kind. In all these cases, the co-occurrence based analysis splits several equivalence classes unnecessarily.

In the following, we consider a more general similarity measure between words for refining stemming algorithms.

4 Distribution Based Stemmer Refinement

We observe that though similar words might not occur in the same document, they are very likely to appear in similar documents of a corpus. So, instead of considering words to be unrelated when they do not co-occur in a single document, two words are considered dissimilar if they do not occur together in a class of documents. Thus, this methodology may be viewed as a generalization of co-occurrence based stemmer refinement. We utilize the information available in a classified text corpus to perform the splitting. The primary assumption behind the proposed methodology is that two words may be stemmed to the same stem if they are extremely similar in their distribution across various categories. This idea is similar to that described in [10].

Each word is assumed to have a multinomial distribution [2] over the set of categories of the given corpus. Words deemed to be arising from the same multinomial distribution are kept in the same equivalence class, whereas, those which are significantly different from each other are separated out. The distribution of each word is estimated from its frequencies in the various categories. Formally, the proposed methodology is as described below.

Let $\{w_1, w_2, \ldots, w_n\}$, be the set of words belonging to an equivalence class, *i.e.*, they all stem to the same stem. Let K be the number of categories of the given text corpus. For each word w_i, we compute the occurrence vector

$n_{i1}, n_{i2}, \ldots, n_{iK}$, where n_{ij} is the number of occurrences of w_i under the jth category. We assume that each w_i arises from a multinomial distribution whose parameters are $p_{i1}, p_{i2}, \ldots, p_{iK}$, and $n_i = \sum_{j=1}^{K} n_{ij}$. Here, each p_{ij} denotes the probability of w_i appearing under the jth category and is estimated as the corresponding proportion of occurrences in the corpus, n_{ij}/n_i.

The aim is to partition this set of words into non-empty subsets such that each subset consists of words whose estimated distributions are not significantly different from each other. Moreover, this task needs to be done without a prior knowledge of the size of the partition.

We employ a procedure similar to sequential hypothesis testing [11] for attaining this goal. Two thresholds/cutoffs, say t_1 and t_2, ($t_1 <= t_2$), are chosen for this purpose, and for each given equivalence class, we try to split it as described hereunder. The words are sorted in descending order of their frequencies. Without loss of generality, we shall now denote this sorted list of words by $\{w_1, w_2, \ldots, w_n\}$. The most frequent word, w_1, is chosen and is considered to stem to itself. We denote this as $stem(w_1) = s_1$. Let S be the current set of stems. Right now, $S = \{s_1\}$. We shall also denote the equivalence class of stem s_j by S_j, defined as $S_j = \{w_k : stem(w_k) = s_j\}$.

Let, $(n_{i1}, n_{i2}, \ldots, n_{iK})$ and $(m_{j1}, m_{j2}, \ldots, m_{jK})$ be the topic vectors of w_i and S_j, respectively. Here, m_{jl}, $l \in \{1, 2, \ldots, K\}$, is defined as the number of occurrences of any of the words in S_j under topic l. It is assumed that the estimated distribution of S_j is the actual one. To test if $(n_{i1}, n_{i2}, \ldots, n_{iK})$ has arisen from the distribution of S_j, Pearson statistic is computed as: $\frac{m_j}{n_i} \sum_{l=1}^{K} \frac{n_{il}^2}{m_{jl}} - n_i$, where, n_i and m_j are the totals, as defined above. Since some of the n_{il}'s may be zero, we replace them by $n'_{il} = (1-\alpha)n_{il} + \alpha\frac{n_i}{K} = n_{il} + \alpha\left(\frac{n_i}{K} - n_{il}\right)$. For small values of α, the frequencies, and hence, the test statistics, do not differ by much from those with α set to 0. We set $\alpha = 0.1$ in our experiments.

For a word w_i, and for each stem $s_j \in S$, we compute the Pearson statistic $d_{i,j}$. If each of these values is greater than the bigger cutoff, *i.e.*, $d_{ij} > t_2\ \forall j$, we shall call the current word as a new stem and add it to the set S. On the other hand, if any of the distances, say d_{ij}, is smaller than t_1, we shall add the current word to the equivalence class of s_j, so that $stem(w_i) = s_j$.

If $t_1 < t_2$, there may be some words which were neither merged with an existing class, nor put into a class of their own. Such words would be dealt with again in the next iteration. After each iteration, t_1 is increased and t_2 reduced. In the last iteration, t_1 is chosen to be equal to t_2.

When n_i is large, the Pearson statistic is known to approximately follow a χ^2 distribution with $K-1$ degrees of freedom. Since we have sorted the words in descending order of their frequencies, the $\chi^2_{(K-1)}$ assumption is satisfied initially.

There are no strict guidelines for choosing t_1 and t_2. If $t_1 = t_2$, then we do not need multiple iterations in the given procedure. This would result in a reduction in computing time. However, it may miss out on some simple mergers of equivalence classes. This is so because, once a word is called a new stem, it cannot be merged with any of the existing stems at a later stage. Choosing $t_1 < t_2$ allows us to do just that. In this case, whenever one is sure of neither

merging the current word with an existing stem nor assigning it to a new class of its own, this decision may be put off for later. In a following iteration, due to the change in the structure of the classes or the values of the chosen thresholds, the decision may become clearer. The strength of the stemmer would be proportional to the size of the thresholds t_1 and t_2, because the size of the dictionary reduces (due to more words being grouped together) with increasing thresholds.

5 Experimental Results

The 20 Newsgroups collection [12] is used to demonstrate the manner in which the proposed methodology splits stem classes as compared to that of the co-occurrence analysis based one. The 20 Newsgroups data set is a collection of 19,997 newsgroup documents, partitioned evenly across 20 different newsgroups. After stopword removal and lowercase conversion, the number of distinct words appearing in at least two documents of the corpus is 56436. These were stemmed by both Porter and Truncate(3) algorithms resulting in 40821 and 8158 stem classes, respectively.

Of the 40821 equivalence classes generated by the Porter stemmer, 32479 were singletons. So, only the remaining 8342 stem classes were considered for splitting. These were split into 16023 and 14962 classes by the distribution and co-occurrence based refinements, respectively. The 8158 equivalence classes generated by Truncate(3) have been split into 22643 and 30710 classes by the distribution and co-occurrence based refinements, respectively.

We examined the refinements of Truncate(3) stemmer and our observations are as follows:

- The co-occurrence based refinement resulted in several extremely large classes. For example, classes corresponding to account, accelerate, accident, accomplish, accuse, *etc.*, were all merged into one single class. Similarly, the classes corresponding to war, ware, ward, *etc.* were not separated out. This was referred to as the stringing effect in [13]. The proposed method split all of them into separate classes.
- Some classes were split unnecessarily by the co-occurrence based method. For example, angle, angles, angular, angstrom, *etc.* were all split into different classes. Our method, however, kept them all together.
- Classes containing certain words which are very similar to each other were split by the co-occurrence based method because they seldom occurred in together in the same document. For example, necessary, necessity, necessarily, *etc.* were all separated out into singleton equivalence classes, because documents containing necessity did not contain necessary and necessarily. Our method, which looks beyond just co-occurrences in a document, merged them all into a single class, with the stem being necessary.

Thus, the equivalence classes generated by the proposed refinement procedure are more appropriate than those by co-occurrence based analysis. Moreover, as seen above, our methodology may also result in fewer number of stems at the same time.

6 Conclusions

We have described the design of a stemming algorithm which uses the class information of a corpus to refine a given stemmer. The main advantage over other stemmers like co-occurrence based stemmers is its ability to drastically reduce the dictionary size. The refined stem equivalence classes are also more appropriate in comparison to those generated by alternative methods. These qualitative results need to be measured quantitatively in terms of precision and recall for retrieval tasks and accuracy for text classification tasks.

Acknowledgment

B. L. Narayan gratefully acknowledges the ISI-INSEAD (France) Fellowship to carry out his doctoral research. This work was also partially supported by CSIR Grant No. 22(0346)/02/EMR-II.

References

1. Frakes, W.B., Fox, C.J.: Strength and similarity of affix removal stemming algorithms. ACM SIGIR Forum **37** (2003) 26–30
2. Johnson, N.L., Kotz, S., Balakrishnan, N.: Discrete Multivariate Distributions. Wiley-Interscience (1997)
3. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18** (1975) 613–620
4. Kraaij, W., Pohlmann, R.: Viewing stemming as recall enhancement. In Frei, H.P., Harman, D., Schauble, P., Wilkinson, R., eds.: Proceedings of the 17th ACM SIGIR conference, Zurich (1996) 40–48
5. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag, New York (1995)
6. Porter, M.F.: An algorithm for suffix stripping. Program **14** (1980) 130–137
7. Krovetz, R.: Viewing morphology as an inference process. In Korfhage, R., Rasmussen, E., Willett, P., eds.: Proceedings of the 16th ACM SIGIR conference, Pittsburgh (1993) 191–202
8. Paice, C.D.: A method for the evaluation of stemming algorithms based on error counting. Journal of the American Society for Information Science **47** (1996) 28–40
9. Yamout, F., Demachkieh, R., Hamdan, G., Sabra, R.: Further enhancement to Porter algorithm. In: Proceedings of the KI2004 Workshop on Machine Learning and Interaction for Text-based Information Retrieval, Germany (2004) 7–24
10. Pereira, F., Tishby, N., Lee, L.: Distributional clustering of English words. In: 31st Annual Meeting of the ACL. (1993) 183–190
11. Wald, A.: Sequential Analysis. Wiley and Sons, New York (1947)
12. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.
13. Xu, J., Croft, W.B.: Corpus-based stemming using coocurrence of word variants. ACM Transactions on Information Systems **16** (1998) 61–81