Corpus-Specific Stemming using Word Form Co-occurrence

W. Bruce Croft and Jinxi Xu
Computer Science Department
University of Massachusetts, Amherst

Abstract

Stemming is used in many information retrieval (IR) systems to reduce word forms to common roots. It is one of the simplest and most successful applications of natural language processing for IR. Current stemming algorithms are, however, either inflexible or difficult to adapt to the specific characteristics of a text corpus, except by the manual definition of exception lists. We propose a technique for using corpus-based word co-occurrence statistics to modify a stemmer. Experiments show that this technique is effective and is very suitable for query-based stemming.

1 Introduction

Stemming is a common form of language processing in most information retrieval systems [4]. It is similar to the morphological processing used in natural language processing, but has somewhat different aims. In an information retrieval system, stemming is used to reduce different word forms to common roots, and thereby improve the ability of the system to match query and document vocabulary. The variety in word forms comes from both inflectional and derivational morphology and stemmers are usually designed to handle both, although in some systems stemming consists solely of handling simple plurals. Stemmers have also been used to group or conflate words that are synonyms (such as "children" and "childhood"), rather than variant word forms, but this is not a typical function. Although stemming has been studied mainly for English, there is evidence that it is useful for a number of languages.

Stemming in English is usually done during document indexing by removing word endings or suffixes using tables of common endings and heuristics about when it is appropriate to remove them. One of the best-known stemmers used in experimental IR systems is the Porter stemmer [5], which iteratively removes endings from a word until termination conditions are met. The Porter stemmer has a number of problems. It is difficult to understand and modify. It makes errors by being too aggressive in conflation (e.g. "policy"/"police", "execute"/"executive" are conflated) and by

missing others (e.g. "European"/"Europe", "matrices"/"matrix" are not conflated). It also produces stems that are not words and are often difficult for an end user to interpret (e.g. "iteration" produces "iter" and "general" produces "gener"). Despite these problems, recall/precision evaluations of the Porter stemmer show that it gives consistent (if rather small) performance benefits across a range of collections, and that it is better than most other stemmers.

Krovetz [4] developed a new approach to stemming based on machine-readable dictionaries and well-defined rules for inflectional and derivational morphology. This stemmer (now called KSTEM) addresses many of the problems with the Porter stemmer, but does not produce consistently better recall/precision performance. One of the reasons for this is that KSTEM is heavily dependent on the entries in the dictionary being used, and can be conservative in conflation. For example, because the words "stocks" and "bonds" are valid entries in a dictionary for general English, they are not conflated with "stock" and "bond", which are separate entries. If the database being searched is the Wall St. Journal, this can be a problem.

The work reported here is motivated by two ideas; corpus-specific stemming and query-based stemming. Corpus-specific stemming refers to automatic modification of conflation classes (words that result in a common stem or root) to suit the characteristics of a given text corpus. This should produce more effective results and less obvious errors from the end user's point of view. The basic hypothesis is that word forms that should be conflated for a given corpus will co-occur in documents from that corpus. Based on that hypothesis, we use a co-occurrence measure similar to the expected mutual information measure (EMIM [8, 1]) to modify conflation classes generated by the Porter stemmer.

In query-based stemming, all decisions about word conflation are made when the query is formulated, rather than at document indexing time. This greatly increases the flexibility of stemming and is compatible with corpus-specific stemming in that explicit conflation classes can be used to expand the query.

In the next two sections, we present these ideas in more detail. In section 4, we discuss the specific corpora we used in the experiments and give examples of the conflation classes that are generated and how they are modified. Section 5 gives the results of retrieval tests that were done with the new stemming approach.

2 Corpus-Specific Stemming

General-purpose language tools have generally not been successful for IR. For example, using a general thesaurus for automatic query expansion does not improve the effectiveness of the system and can, indeed, result in less effective retrieval (e.g. [9]). When the tool can be tuned to a given domain or text corpus, however, the results

are usually much better[1].

From this point of view, stemming algorithms have been one of the more successful general techniques in that they consistently give small effectiveness improvements. In most applications where an IR system includes a stemmer, exception lists are used to describe conflations that are of particular importance to the application, but are not handled appropriately by the stemmer. For example, an exception list may be used to guarantee that "Japanese" and "Chinese" are conflated to "Japan" and "China" for an application containing export reports. Exception lists are constructed manually for each application. Using human judgement for these lists is expensive and can be inconsistent in quality.

Instead of the manual approach of exception lists, the conflations performed by the stemmer can be modified automatically using corpus-based statistics. To do this, we assume that word forms that should be conflated will occur in the same documents or, more specifically, in the same text windows. For example, articles from the Wall St. Journal that discuss stock prices will typically contain both the words "stock" and "stocks". This technique should identify words that should be conflated but are not ("stock" and "stocks" are an example for KSTEM), and words that should not be conflated but are. Examples of the latter are the word pairs "policy"/"police" and "addition"/"additive" for the Porter stemmer.

The basic measure that is used to measure the significance of word form co-occurrence is a variation of EMIM [8, 1]. This measure is defined for a pair of words a and b by the following formula:

$$em(a,b) = \frac{n_{ab}}{n_a + n_b}$$

where n_a, n_b are the number of occurrences of a and b in the corpus, and n_{ab} is the number of times both a and b fall in a text window of size win in the corpus. We define n_{ab} as the number of elements in the set $\{< a_i, b_j > | dist(a_i, b_j) < win\}$, where a_i's and b_j's are distinct occurrences of a and b in the corpus, and $dist(a_i, b_j)$ is the distance between a_i and b_j measured using a word count within each document.

Given this measure, the question is which word pair statistics should be measured? In previous studies, the EMIM measure has been applied to all word pairs that co-occur in text windows. The aim of this type of study was to discover phrasal and thesuarus relationships. In this study, we have a different aim, namely, to clarify the relationship between words that have similar morphology. For this reason, the em measure is calculated only for word pairs that potentially could be conflated. The way we have chosen to do this is to use an "aggressive" stemmer (Porter) to identify words that may be conflated, and then use the corpus statistics to refine that conflation.

[1] Jing and Croft [3] discuss a corpus-based technique for query expansion that produces significant effectiveness improvements

A problem with this approach is that if the aggressive stemmer is not aggressive enough, word pairs that should be conflated will be missed. There are a number of ways that this could be addressed, such as identifying word pairs with significant trigram overlap. In our work, we have combined the Porter stemmer with KSTEM to identify possible conflations. Even though KSTEM is not as aggressive as Porter, it does conflate some words that Porter does not. For example, the Porter stemmer conflates "abdomen" and "abdomens", but not "abdominal". KSTEM conflates all of these.

More generally, we can view stemming as constructing equivalence classes of words. For example, the Porter stemmer conflates "bonds", "bonding", and "bonded" to "bond", so these words form an equivalence class. The corpus statistics for word pairs in the equivalence classes are used to determine the final classes. For example, if "bonding" and "bonded" do not co-occur significantly in a particular corpus, one or both of them may be removed from the equivalence or conflation class, depending on their relationship to the other words.

More specifically, if a and b are stemmed to c, then all occurrences of a, b and c are the same after the stemming transformation, i.e. a, b and c form an equivalence class. If, however, a is stemmed to b, and b is stemmed to c, then a, b and c do not form an equivalence class. The Porter stemmer occasionally makes such incomplete conflations. We consider this a "bug" of the Porter stemmer, and put a, b and c in an equivalence class.

Suppose the collection has a vocabulary $V = \{w_1, w_2, ..., w_n\}$. We use the union-find algorithm to construct the equivalence classes as follows:

1. For each word w_i, use the Porter stemmer to stem it to r_i. Let $R = \{r_i\}$.
2. For each element in $V \bigcup R$ form a singleton class.
3. For each pair $< w_i, r_i >$, if w_i and r_i are not in the same equivalence class, merge the two equivalence classes into one.
4. In each equivalence class, remove those elements not in V.

The union find algorithm runs in $O(n \log^* n)$, "almost" linear time because $log^* n$ is a small number even if n is very large.

Given the final equivalence classes, a representative or stem for each class must be generated. We chose simply to use the shortest word in the class. As well as being simple, this has the desirable result of producing complete words instead of the usual type of Porter stem.

The other issue is what to do with word pairs for which there is insufficient statistics. If the words in a conflation class are rare in the corpus, the *em* measure will be unreliable. For these pairs, we chose to use KSTEM to determine whether they

should remain in an equivalence class. The threshold used for sufficient statistics is that $n_a + n_b > 50$.

To summarize, the overall process for producing corpus-specific conflation classes consists of the following steps:

1. Collect the unique words (the vocabulary V) in the corpus. This is done using a simple *flex* scanner. Numbers, stop words and possible proper nouns are discarded.

2. Construct equivalence classes using the Porter stemmer, sometimes augmented by KSTEM.

3. Calculate em for every pair of words in the same equivalence class.

4. Form new equivalence classes. This is done by starting with every word in V forming a singleton equivalence class. Then every em pair, if $em(a, b) > min$ and they are not in the same class, merge the equivalence classes. If the statistics are inadequate, use KSTEM to decide whether to merge classes.

5. Make a stem dictionary from the equivalence classes for future use in indexing and query processing. The shortest word in each class is the "stem" for that class.

Timing figures and class statistics for sample corpora are presented in section 4.

3 Query-Based Stemming

The corpus-based stemming approach described in the last section produces a dictionary of words with the appropriate stem. Given this dictionary, the usual process of stemming during indexing can be replaced by dictionary lookup. Alternatively, the full word form could be used for indexing and stemming would become part of query processing. The way this would work is that when a query is entered, the equivalence class for each non-stopword would be used to generate an expanded query. For example, if the original query (in the INQUERY query language) was

```
#SUM(stock prices for IBM)
```

The expanded query for a particular corpus could be

```
#SUM( #SYN(stock stocks) #SYN(price prices) IBM)
```

The #SYN operator is used to group synonyms. Depending on the details of how this is done in the underlying system, this query will produce the same result as a query processed in an environment where the database had been indexed by stems.

The advantages of query-based stemming are that the user of the system can be consulted as to the applicability of particular word forms and queries can be restricted to search for a specific word form. These advantages can be significant in cases where small differences in word forms result in large differences in the relevance of the retrieved documents. For example, in looking for articles about terrorist incidents, the word "assassination" is very good at discriminating relevant from non-relevant documents, but the word "assassinations" is much less useful [6].

The main disadvantage of query-based stemming is that the queries become longer and will, therefore, take longer to process. The impact on response times will depend on the degree of query expansion. In the next section, we present statistics for some corpora.

4 Corpora and Conflation Classes

The corpora that we use in these experiments are the West collection of law documents and the Wall St. Journal collection of newspaper articles [7, 2]. The statistics for these corpora and the associated queries and relevance judgements are shown in Table 1. Two sets of queries are used for the West collection. The first is where the queries are treated as a collection of individual words. The second uses INQUERY operators (such as #PHRASE) to structure the combinations of words. In previous work, stemming on phrasal units can produce different results than word-only stemming. Retrieval results for both types of query are presented in the next section.

	WEST	**WSJ**
Number of queries	34	66
Number of documents	11,953	162,795
Mean words per query	9.6	37.5
Mean words per document	3,262	260
Mean relevant documents per query	28.9	144
Number of words in a collection	39,000,000	42,307,309

Table 1: Statistics on text corpora

As an example of the timing figures for generating conflation classes, the following figures are for the WSJ corpus. All timing figures are CPU times for a SUN4

workstation. To collect the unique words in the corpus takes 20 minutes. There were 76,181 of these. The Porter stemmer takes 6 seconds to stem these words and the union-find algorithm takes 9 seconds to form equivalence classes. The number of classes produced is 39,225 and the average class size is 1.96 word forms. Generating the *em* values for a text window of size 100 words ($win = 100$), takes 1 hour and 10 minutes. With a threshold for the *em* value of .01 ($min = 0.01$), the number of conflation classes generated is 65,104 with an average class size of 1.17. Using these classes as the basis for stemming produces the best retrieval results (shown in the next section) and avoids 70% of the conflations made by Porter. This means that query expansion is reduced in a query-based stemming environment.

For the West collection, there are 49,964 unique words which generate 27,117 classes using the Porter stemmer. After application of the *em* threshold, there were 40,012 classes.

As an example, with these classes, "bonds" is conflated to "bond" and "bonding" is conflated to "bonded" in the WSJ corpus. In the West corpus, all words are conflated to "bond".

Figure 1 contains examples of the conflation classes for Porter on the WSJ corpora and Figure 2 has the classes after application of the *em* threshold.

```
abandon abandoned abandoning abandonment abandonments abandons
abate abated abatement abatements abates abating
abrasion abrasions abrasive abrasively abrasiveness abrasives
absorb absorbable absorbables absorbed absorbencies absorbency absorbent
  -absorbents absorber absorbers absorbing absorbs
abusable abuse abused abuser abusers abuses abusing abusive abusively
access accessed accessibility accessible accessing accession
```

Figure 1: Example of conflation classes on WSJ using Porter

The *em* value depends on the window size. The larger the window, the higher the *em* values that are generated. When the window size is fixed, the *em* threshold controls how any conflations by Porter are prevented. By experimenting with different window sizes and threshold values, we found that as long as a reasonable sized window (larger than 50) is used, performance depends only on the percentage of conflations that are prevented.

The dominant overhead of our method is the time to collect co-occurrence data. This is proportional to the window size. Since performance does not directly depend on window size, a 100 word window represents a good compromise between performance and computational overhead.

```
abandonment abandonments
abated abatements abatement
abrasive abrasives
absorbable absorbables
absorbencies absorbency absorbent
absorber absorbers
abuse abusing abuses abusive abusers abuser abused
accessibility accessible
```

Figure 2: Example of conflation classes on WSJ after co-occurrence thresholding

5 Retrieval Results

The following tables give standard recall/precision results for retrieval experiments carried out using the Porter, KSTEM and *em* modified Porter stemmers (NEW). Table 2 shows the results of the simple word-based queries on the West corpus. The results show little difference between the stemmers, with perhaps a small advantage at higher recall levels for the NEW stemmer. Table 3 gives the results for the phrase-based queries for West. These results give an advantage to both Porter and the NEW stemmers.

Table 4 gives the results for the WSJ collection. We see again a clear advantage for the Porter and NEW stemmers. Overall, the NEW stemmer has very consistent performance and may be able to combine the advantages of both the Porter and KSTEM approaches.

The final experiment, shown in Table 5, uses KSTEM to decide whether a word pair is conflated when there is insufficient statistics. Comparing this table to the previous one, we see there is little difference. Words that do not occur frequently enough to generate reliable *em* values are unlikely to affect retrieval on an average basis. For individual queries, however, this modification could be very important.

6 Conclusions

A new approach to stemming that uses corpus-based statistics was proposed. This approach can potentially avoid making conflations that are not appropriate for a given corpus and uses an "aggressive" stemmer as a starting point. The result of this stemmer is an actual word rather than an incomplete stem, as is often the case with the Porter approach. It can also be implemented efficiently and is suitable for query-based stemming. The experimental results show that the new stemmer gives more

consistent performance improvements than either the Porter or KSTEM approaches.

Acknowledgements

This work was supported by the NSF Center for Intelligent Information Retrieval at the University of Massachusetts. Bob Krovetz helped with the organization of the experiments.

References

[1] K. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th ACL Meeting*, pages 76–83, 1989.

[2] D. Harman. Overview of the first TREC conference. In *Proceedings of the 16th ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 36–47, 1993.

[3] Y. Jing and W.B. Croft. An association thesaurus for information retrieval. In *Proceedings of RIAO 94*, 1994. to appear.

[4] Robert Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th International Conference on Research and Development in Information Retrieval*, pages 191–202, 1993.

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

[6] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems*, 12:296–333, 1994.

[7] Howard Turtle. Natural language vs. Boolean query evaluation: A comparison of retrieval performance. In *Proceedings ACM SIGIR 94*, pages 212–220, 1994.

[8] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, second edition, 1979.

[9] E. Voorhees. Query expansion using lexical-semantic relations. In *Processings of the 17th ACM SIGIR Conference*, pages 61–69, 1994.

Recall	Precision (34 queries)		
	KSTEM	PORTER	NEW(100-0.01)
10	79.2	78.0 (-1.5)	78.0 (-1.5)
20	75.7	73.7 (-2.6)	75.3 (-0.6)
30	71.7	71.9 (+0.2)	72.6 (+1.3)
40	61.9	61.8 (-0.1)	62.0 (+0.2)
50	54.8	54.8 (+0.1)	54.8 (+0.1)
60	46.2	45.0 (-2.6)	46.2 (+0.1)
70	37.4	37.0 (-1.1)	38.2 (+2.2)
80	29.0	29.7 (+2.6)	31.6 (+9.0)
90	16.7	17.9 (+7.3)	18.3 (+9.6)
100	9.1	10.6 (+16.4)	10.0 (+10.0)
avg	48.2	48.0 (-0.2)	48.7 (+1.1)

Table 2: Retrieval experiments on the West corpus

Recall	Precision (34 queries)		
	KSTEM	PORTER	NEW(100-0.01)
10	79.2	79.7 (+0.6)	79.3 (+0.1)
20	75.6	74.5 (-1.4)	75.0 (-0.8)
30	71.8	71.1 (-1.0)	71.9 (+0.1)
40	63.9	63.8 (-0.2)	63.8 (-0.2)
50	58.1	58.4 (+0.5)	60.0 (+3.3)
60	50.7	51.3 (+1.1)	52.0 (+2.5)
70	41.9	42.2 (+0.8)	42.6 (+1.6)
80	32.6	34.4 (+5.6)	34.9 (+7.0)
90	20.5	21.4 (+4.1)	21.8 (+6.1)
100	10.3	11.4 (+10.6)	10.5 (+1.7)
avg	50.5	50.8 (+0.7)	51.2 (+1.4)

Table 3: Retrieval experiments using West structured queries

Recall	Precision (66 queries)		
	KSTEM	PORTER	NEW(100-0.01)
10	52.3	52.7 (+0.7)	52.8 (+0.9)
20	45.1	45.4 (+0.6)	45.8 (+1.6)
30	39.4	40.8 (+3.5)	41.3 (+4.8)
40	35.1	36.2 (+3.1)	36.8 (+5.0)
50	29.9	31.1 (+4.1)	31.3 (+4.6)
60	24.9	26.3 (+5.8)	26.3 (+5.5)
70	20.8	22.0 (+5.5)	22.0 (+5.3)
80	16.5	17.2 (+4.5)	17.3 (+5.1)
90	11.4	12.0 (+5.5)	12.1 (+6.7)
100	2.5	2.8 (+11.7)	2.8 (+15.5)
avg	27.8	28.6 (+3.1)	28.9 (+3.9)

Table 4: Retrieval experiments using WSJ corpus

Recall	Precision (66 queries)	
	KSTEM	NEW(100-0.01)
10	52.3	52.8 (+0.8)
20	45.1	45.8 (+1.5)
30	39.4	41.3 (+4.8)
40	35.1	36.8 (+5.0)
50	29.9	31.3 (+4.7)
60	24.9	26.3 (+5.5)
70	20.8	22.0 (+5.3)
80	16.5	17.3 (+5.0)
90	11.4	12.1 (+6.7)
100	2.5	2.8 (+15.5)
avg	27.8	28.8 (+3.8)

Table 5: Retrieval experiments using KSTEM for pairs with insufficient statistics (WSJ collection)