IR-95

Corpus-Based Stemming using Co-occurrence of Word Variants

Jinxi Xu and W. Bruce Croft

Computer Science Department

University of Massachusetts, Amherst, MA 01003. {xu, croft}@cs.umass.edu

Abstract

Stemming is used in many information retrieval (IR) systems to reduce variant word forms to common roots. It is one of the simplest applications of natural language processing to IR, and one of the most effective in terms of user acceptance and consistent, though small, retrieval improvements. Current stemming techniques do not, however, reflect the language use in specific corpora and this can lead to occasional serious retrieval failures. We propose a technique for using corpus-based word variant co-occurrence statistics to modify or create a stemmer. The experimental results generated using English newspaper and legal text and Spanish text demonstrate the viability of this technique and its advantages relative to conventional approaches.

Categories and Subject Descriptors: H.3.1. [**Information Storage and Retrieval**]: Content Analysis and Indexing – *indexing methods; linguistic processing*; H.3.3. [**Information Storage and Retrieval**]: Information Search and Retrieval – *query formulation; search process*

General terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: information retrieval, stemming, corpus analysis, co-occurrence,

class refinement, n-gram

1 Introduction

Stemming is a common form of language processing in most information retrieval systems [Krovetz 1993]. 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 variant 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.

Evaluations of stemming using test collections have produced mixed results [Harman 1991], but more recent work has shown consistent (if rather small) improvements in retrieval effectiveness across a range of collections [Krovetz 1993; Hull 1996]. Stemming is usually viewed as a recall-enhancing device, since it expands the original query with related word forms, but it can sometimes improve precision at low recall levels by promoting relevant documents to high ranks.

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 [Porter 1980], which iteratively removes endings from a word until termination conditions are met. The Porter stemmer has a number of problems that are found, to varying degrees, in other stemmers.

- It is difficult to understand and modify.

- It makes errors by sometimes 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 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 have shown that it performs at least as well as other stemmers [Hull 1996].

Krovetz [Krovetz 1993] 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, as discussed later.

We refer to the group of words that result in a common root after applying a specific stemming algorithm as a conflation or equivalence class. One such class may be, for example, "stock", "stocks", "stocked", and "stocking". Corpus-based stemming refers to automatic modification of equivalence classes 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 the 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 [van Rijsbergen 1979; Church and Hanks 1989]) to modify an initial set of conflation classes generated by a stemmer. We also test the viability of building a new stemmer without linguistic

knowledge by modifying equivalence classes generated by a simple n-gram approach.

The equivalence classes generated by stemmers and corpus analysis could be used at document indexing time, or when the query is formulated. Query-based stemming is more flexible in that users can choose which word variants are applicable for their query, by eliminating or adding variants to an expanded form of the query. The price for this additional flexibility is efficiency. Expanded queries can take significantly longer to process and the degree of expansion that results from the set of equivalence classes is an important efficiency measure for a stemming algorithm.

In this article, we first describe algorithms for corpus analysis of word variants, and then report on the results of effectiveness and efficiency experiments. All experiments were carried out using the INQUERY retrieval system [Broglio et al. 1994].

2 Corpora Used in Experiments

The retrieval experiments in this study are carried out on three English corpora and a Spanish corpus. The English corpora are the WEST legal document collection [Turtle 1994], and two collections of news articles from the TREC experiments [Harman 1995], WSJ (Wall Street Journal 1987–1991)[1], and WSJ91 (Wall Street Journal 1991). The Spanish corpus is the ISM collection of the TREC conferences. The query set on ISM is the Spanish adhoc queries of TREC 4. Statistics derived from the corpora and the associated queries are shown in table 1. Two sets of queries are used on WEST, a natural language query set and a structured query set. The first is where the queries are treated as a collection of individual words. The second uses operators to structure the combinations of words. In previous work, stemming phrases produced different results than stemming words [Krovetz 1993].

[1]The co-occurrence data for WSJ is actually from Wall Street Journal 1987–1992. Because Wall Street Journal 1992 has only 0.03 GB of data, this fact should not affect the results reported in this paper

3 Corpus Analysis of Word Variants

3.1 Why Corpus Analysis

General-purpose language tools have generally not been successful for IR. For example, using a general thesaurus for automatic query expansion does not consistently improve the effectiveness of the system and can, indeed, result in less effective retrieval (e.g. [Voorhees 1994]). When the tool can be tuned to a given domain or text corpus, however, the results are usually much better[2].

From this point of view, stemming algorithms have been one of the more successful general techniques in that they consistently give small effectiveness improvements. Even with stemming, an algorithm that adapts to the language characteristics of a domain as reflected in a corpus should do better than a non adaptive one. Many words have more than one meaning, but their meanings are not uniformly distributed across all corpora. For example, "stocks" is the plural form of "stock" in Wall Street Journal, but its primary meaning in a corpus about medieval history may be a device to punish prisoners. Therefore, a good conflation for one corpus may be bad for another. Corpus-based statistics may help establish the relationship between words in the specific corpus.

The basic assumption in this paper is that word variants 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 the corpus-dependent conflations ("stock" and "stocks"). It should also help prevent the bad conflations made by a linguistics based stemmer, e.g, "policy"/"police" and "addition"/"additive" by the Porter stemmer, because such unrelated words should co-occur rarely.

[2]Jing and Croft [Jing and Croft 1994] discuss a corpus-based technique for query expansion that produces significant effectiveness improvements

3.2 Metrics

We define the following metric to measure the significance of word form co-occurrence

$$em(a,b) = max(\frac{n_{ab} - En(a,b)}{n_a + n_b}, 0)$$

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 win word tokens in the corpus. More strictly, 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. $En(a,b)$ is the expected number of co-occurrences assuming a and b are statistically independent.

The metric is a variation of EMIM (expected mutual information measure) [van Rijsbergen 1979; Church and Hanks 1989], which is widely used to measure significance of associations. For word form co-occurrences, EMIM can be defined as

$$EMIM(a,b) = P(a,b) log_{10}(\frac{P(a,b)}{P(a)P(b)})$$

where $P(a,b) = n_{ab}/N$, $P(a) = n_a/N$, $P(b) = n_b/N$, N is the number of text windows.

The reason we choose not to use EMIM is that it is not normalized over the number of occurrences of a and b and unfairly favors high frequency word forms. The em metric, however, measures the percentage of the occurrences of a and b which are co-occurrences and intuitively is more suitable for our purpose.

The role of $En(a,b)$ in the em formula is important, because two words may co-occur by chance. For example, the words "the" and "of" almost always co-occur in a reasonably large text window,

but it is erroneous to conclude they are related. We use the the following formula to calculate $En(a,b)$:

$$En(a,b) = kn_a n_b$$

where k is a constant factor given the corpus and the window size. To justify the formula, we make the assumption that a word occurs in a text window at most once. Thus,

$$En(a,b) = P(a,b)N = P(a)P(b)N = \frac{n_a}{N}\frac{n_b}{N}N = \frac{n_a n_b}{N}$$

In practice, we currently estimate k based on the co-occurrence data for a large sample of randomly chosen word pairs by using the value $\sum n_{ab} / \sum n_a n_b$. On WSJ, we used 5000 randomly chosen word pairs and obtained $k = 2.74e^{-6}$ with window size 100 words.

In table 2, we list a number of pairs of words and their *em* scores on WSJ with text window size 100. The table clearly indicates the relationship between *em* score and appropriateness for conflations. Some of the examples are very interesting. "Gases" is the plural form of "gas" and they seem to be closely related. But they have a low *em* score on WSJ. By examining the documents, we find that "gases" usually mean "inert gases", "hot gases" or "medical gases", etc, while "gas" almost always means "natural gas" or "gasoline". "News" and "new" co-occur frequently and Porter stemmer conflates "news" to "new". But according to the formula for estimating $En(a,b)$, their expected number of co-occurrences is 50,000. This results in an *em* score of 0, which suggests they are not related.

corpus	WEST	WSJ	WSJ91	ISM
Number of queries	34	66	60	25
Raw text size in gigabytes	0.26	0.5	0.146	0.2
Number of documents	11,953	163,092	42,652	57,868
Mean words per query	9.6	37.5	35	5.3
Mean words per document	3,262	273	291	283
Mean relevant documents per query	28.9	144	55.6	88
Number of words in a collection	39,000,000	44,495,324	12,418,568	16,427,826

Table 1: Statistics on text corpora

word	frequency	word	frequency	co-occ	em
bond	42255	bonds	49331	37706	0.35
stock	144076	stocks	35898	46030	0.18
cruise	1253	cruises	191	239	0.17
animation	172	animators	29	28	0.14
brokerage	7802	brokers	7191	1890	0.12
votes	3349	voting	4577	625	0.074
gas	20013	gases	419	147	0.006
policy	26122	police	7290	294	0.0
new	225064	news	81711	27307	0.0
arm	3004	army	7684	37	0.0
desirable	681	desires	211	0	0.0

Table 2: em scores of example word pairs on WSJ with 100 word window

3.3 Connected Component Algorithm

Stemming is equivalent to expanding query words by their equivalence classes. An aggressive stemmer generates large equivalence classes by over-stemming. The approach we adopt in this paper is to use corpus analysis based on the *em* score to break up (or refine) the large equivalence classes generated by an aggressive stemmer.

The first algorithm we used in the paper for equivalence class refinement is the connected component algorithm. To refine an initial equivalence class, the class is mapped to a graph, each vertex representing a word and an edge between two vertices representing an *em* score greater than a threshold t. The vertices in each connected component of the graph form a new equivalence class. The threshold t is a parameter in our experiments.

The connected component algorithm is efficient. Our implementation of the algorithm runs in $O(nlog^*n)$ to refine a class of size n, almost linear because log^*n is a small value for any practically large n. But the connected component algorithm occasionally produces large sparsely connected equivalence classes, called "strings" in the clustering literature [Salton 1989], which hurt the retrieval effectiveness. Figure 1 gives some equivalence classes generated by the Porter stemmer on WSJ. The classes that result from applying the connected component algorithm are shown in figure 2 with singleton classes omitted.

3.4 Optimal Partition Algorithm

The other algorithm we used for class refinement is the optimal partition algorithm, which addresses the "string" problem associated with the connected component algorithm. Stemming may improve recall by retrieving more relevant documents. It may also decrease precision by retrieving non-relevant ones. The motivation behind the optimal partition algorithm is to achieve the best tradeoff between the two. More precisely, for any pair of words a and b, we let the recall benefit of keeping a

and b together be $em(a,b)$ because the higher the em score, the more likely that a and b are related and the conflation may improve recall. We let the harm to precision of keeping a and b together be a constant δ. Thus the net benefit of keeping a and b together is $em(a,b) - \delta$ and the net benefit of separating them is 0. The net benefit of a partition of an equivalence class is the sum of the net benefits for all pairs of words in the class. An optimal partition of an equivalence class is one that maximizes the net benefit. If there are several optimal partitions, one of them is chosen arbitrarily to refine the initial equivalence class.

So far we have not found an efficient implementation of the optimal partition algorithm. Its similarity to some NP-complete problems prompts us to guess that it may also be NP-complete. Our current implementation of the algorithm is a brute force search of all possible partitions. The following equations give the number of possible partitions $f(n)$ for a class of n words:

$$f(n) = \sum_{i=0}^{i=n-1} f(n-i-1) \begin{pmatrix} n-1 \\ i \end{pmatrix}$$

$$f(0) = 1$$

$f(n)$ obviously grows exponentially with n(f(10)=115975, f(12)=4213597 and f(14)=190899099). To make it feasible, we first apply the connected component algorithm to make the input classes smaller. If a class is still too large (with more than 12 members), we further split it based on the initial characters of words. Then we apply the optimal partition algorithm. The solution may be suboptimal, but should be adequate for our purpose.

Figure 3 shows the power of the optimal partition algorithm. All the words are clustered in the same connected component equivalence class because they are connected, though sparsely. Each area enclosed in a dotted line is a new equivalence class generated by the optimal partition algorithm. Those words about human race are separated from those words about horse race.

```
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 equivalence classes on WSJ using Porter

```
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: Equivalence classes produced by the connected component algorithm

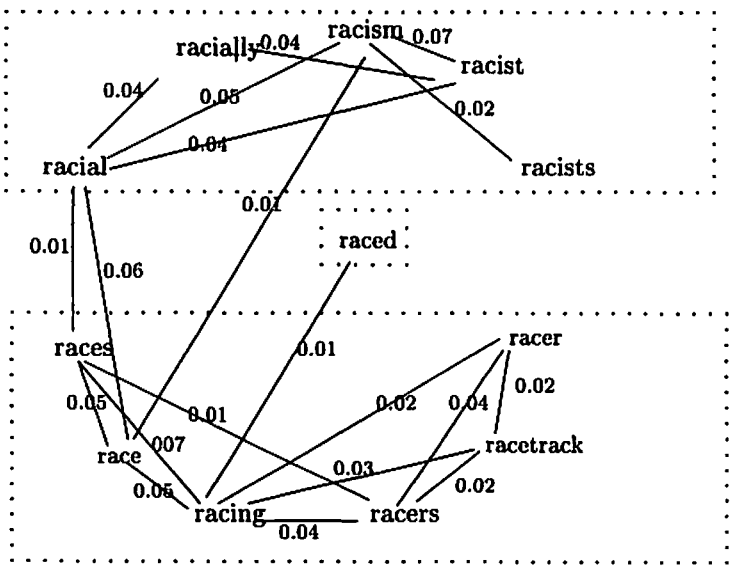

Figure 3: Optimal partition of a connected component equivalence class

4 Experimental Setup

The general procedure for the experiments in this paper is as follows:

1. Collect all unique word forms in the corpus. Numbers, stop words and possible proper nouns are discarded.

2. Construct the initial equivalence classes using an aggressive stemmer.

3. Collect the co-occurrence data for word pairs in the same initial equivalence class and the sample of word pairs for calculating k. Estimate k and calculate the *em* scores for word pairs in the same initial equivalence class.

4. Refine the initial equivalence classes based on the *em* scores using the connected component or/and the optimal partition algorithms.

5. Use the refined equivalence classes for query expansion: a query word is replaced by its equivalence class. The words in the equivalence class are grouped together by the INQUERY #SYN operator so that at retrieval time INQUERY creates a single inverted list for the equivalence class by merging the inverted lists for the words in the class. The expanded queries are run against the unstemmed database of the corpus. Retrieval results are evaluated using the standard 10 point average precision and compared with the queries expanded by the baseline stemmers.

Two aggressive stemmers are used in step 2: the Porter stemmer on the English corpora and a trigram matching algorithm on English and Spanish Corpora. Two words are in the same trigram equivalence class if they share the first three characters. The purpose of using a trigram "stemmer" is twofold. One is test the feasibility of constructing a statistical stemmer with none or little linguistic knowledge. The other is to test the effectiveness of statistical corpus analysis. Because we know

trigram matching will make many mistakes, if corpus analysis can improve its performance to a level comparable with the traditional stemmers, it should validate the value of corpus analysis.

We must choose an appropriate size for the text window that determines co-occurrence in step 3. If the window is too small, the *em* scores for infrequent words (which dominate the vocabulary according to Zipf's Law) will be unreliable due to few co-occurrences. If the window size is too large, step 3 will consume too much time because its running time is proportional to the window size. Our previous work showed that the appropriate window size should be 50–100 words [Croft and Xu 1995]. In this work, we always use 100 as the window size.

In step 4, we must decide on the *em* threshold for the connected component algorithm and the δ value for the optimal partition algorithm. When the window size is fixed, the *em* threshold controls how many conflations are prevented: the higher the threshold, the smaller the equivalence classes after applying the connected component algorithm and the more conservative the resulted stemmer. Our previous work [Croft and Xu 1995] showed that for window size 100, the appropriate *em* threshold is 0.01, which is used in this paper. Like the *em* threshold in the connected component algorithm, the δ value in the optimal partition algorithm controls how conservative is the resulted stemmer: a higher δ value results in a more conservative stemmer. We have tried different δ values and table 13 shows the influence of δ on retrieval effectiveness on WSJ. We found that the appropriate δ value for a 100 word text window is around 0.0075, which is about half the average *em* value between pairs of words in the same initial equivalence classes. In this work, δ is set 0.0075, though a more principled approach which we will take in the future is to set the δ value based on the average *em* score. In our experiment we found that retrieval performance is relatively insensitive to small variations of the parameter values.

5 English Experiments

5.1 Baseline Stemmers

Porter and KSTEM are the two baseline stemmers against which we compare retrieval results on the English Corpora. The retrieval effectiveness of query-based stemming by Porter and KSTEM is shown in tables 3, 4 and 5. KSTEM outperforms Porter on the natural language queries on the WEST corpus, while Porter outperforms KSTEM on WSJ.

Porter and Kstem are quite different in terms of the amount of query expansion. WSJ has 76181 unique word variants. The Porter stemmer clusters them into 39949 equivalence classes, with an average class size of 1.9. KSTEM clusters them into 44543 classes, with average class size 1.7. Though the average class sizes are similar for the two stemmers, the Porter stemmer produces much longer expanded queries than Kstem. We define the expansion factor of a stemmer as the number of words in the expanded query set by the stemmer over the number of words in the unexpanded query set. The expansion factor of the Porter stemmer is 4.5 on WSJ compared to 2.8 of KSTEM.

The WEST corpus has 49,964 unique word variants, and Porter and KSTEM cluster them into 27,117 and 26,211 classes respectively, with average class sizes 1.84 and 1.9. The expansion factor of Porter is 5.5, much higher than the 3.1 value for KSTEM.

The expansion factors show that Porter is a more aggressive stemmer than KSTEM. For example, on WEST, the Porter stemmer expands the word "constitutes" to "constitute constituted constitutent constitutents constitutes constituting constitution constitutional constitutions constitutive constituttional", while KSTEM expands it to "constitute constituted constitutes constituting".

5.2 Refining Porter Equivalence Classes

In this section, we use the connected component and optimal partition algorithms to refine Porter stemmer's equivalence classes. On WSJ, the 39,949 Porter equivalence classes are broken up into 64,821 classes by the connected component algorithm with average class size 1.17, which are further broken up into 73,015 classes of average size 1.04 by the optimal partition algorithm. The expansion factor is reduced from 4.5 to 2.2 to 2.06. On WEST, the 27,117 Porter equivalence classes are broken up into 40,215 classes by the connected component algorithm with average class size 1.24, which are further broken up into 46,632 classes of average size 1.07 by the optimal partition algorithm. The expansion factor is reduced from 5.5 to 2.9 to 2.6.

As an example, the word "constitute" in the WEST queries is expanded to "constitute constituted constitutes" after applying the connected component algorithm, and to "constitute constitutes" after applying the optimal partition algorithm.

Less expansion not only means faster retrieval (see the timing figures in section 5.6), but also results in better retrieval effectiveness. As we see in tables 3, 4 and 5 the connected component and optimal partition algorithms produce small improvements in retrieval effectiveness on both WEST and WSJ. The effectiveness of the optimal partition algorithm is always better than that of the connected component algorithm. In an interactive environment where the expansion words are visible to end users, eliminating the bad conflations also means better user satisfaction.

5.3 Refining Trigram Equivalence Classes

On WSJ, trigram matching generates 2,952 classes from the 76,181 unique word variants with average class size 2.58. While the average class size is not large, some classes are huge: 165 classes have more than 100 members. The largest is the equivalence class with initial characters "con", with 1,124 members. There are over 6 million word pairs whose co-occurrence data we need to collect

	Precision (% change) – 34 queries						
Recall	kstem	porter		porterCC		porterOptimal	
10	79.9	78.2	(−2.1)	79.1	(−1.0)	80.4	(+0.6)
20	75.3	72.8	(−3.3)	74.0	(−1.7)	75.8	(+0.6)
30	71.4	70.1	(−1.8)	71.3	(−0.1)	72.6	(+1.7)
40	61.2	60.2	(−1.7)	60.4	(−1.3)	61.1	(−0.1)
50	54.0	52.9	(−2.0)	54.6	(+1.0)	54.6	(+1.1)
60	44.1	44.5	(+0.9)	45.8	(+4.0)	46.0	(+4.4)
70	36.0	35.5	(−1.5)	37.2	(+3.2)	37.1	(+2.9)
80	27.0	28.7	(+6.4)	30.8	(+14.1)	30.6	(+13.5)
90	15.7	16.5	(+4.9)	16.4	(+4.4)	16.3	(+3.9)
100	8.1	8.8	(+7.9)	9.1	(+12.1)	8.9	(+9.9)
average	47.3	46.8	(−1.0)	47.9	(+1.3)	48.3	(+2.3)

Table 3: Improving Porter stemmer on WEST using natural language queries

	Precision (% change) – 34 queries						
Recall	kstem	porter		porterCC		porterOptimal	
10	80.9	80.7	(−0.2)	80.7	(−0.2)	81.2	(+0.3)
20	75.5	74.6	(−1.1)	74.8	(−0.9)	75.3	(−0.3)
30	72.0	70.7	(−1.9)	71.3	(−1.0)	72.1	(+0.1)
40	63.8	63.7	(−0.2)	64.0	(+0.4)	64.5	(+1.1)
50	57.2	57.4	(+0.3)	58.6	(+2.5)	58.6	(+2.5)
60	49.2	49.7	(+0.9)	50.5	(+2.7)	50.4	(+2.4)
70	41.5	41.9	(+1.0)	42.1	(+1.3)	42.2	(+1.6)
80	30.7	33.4	(+8.7)	33.4	(+8.6)	33.3	(+8.5)
90	19.7	20.1	(+2.3)	20.4	(+3.7)	20.4	(+3.8)
100	9.3	9.1	(−1.7)	9.4	(+1.2)	9.4	(+1.2)
average	50.0	50.1	(+0.3)	50.5	(+1.1)	50.7	(+1.5)

Table 4: Improving Porter stemmer on WEST using structured queries

from WSJ, but only 278,459 pairs ever co-occur in the corpus.

Using the connected component algorithm, these classes are refined into 57,295 classes, with average class size 1.32. Some classes are still too large and contain words not morphologically related, e.g., "company" and "computer". To address this problem, more initial characters are used to determine the association between common prefixed words. If two words have the same prefix but differ in the next 3 characters, their *em* score is forced to be 0. Thus "company" and "computer" have *em* value 0 because "pan" and "put" are different. Because it may use more than 3 characters, we label this modified approach *n-gram*.

	Precision (% change) – 66 queries						
Recall	kstem	porter		porterCC		porterOptimal	
10	51.7	53.0	(+2.6)	52.7	(+2.0)	53.0	(+2.6)
20	45.0	45.2	(+0.5)	45.8	(+1.9)	45.9	(+1.9)
30	39.2	40.5	(+3.3)	40.8	(+4.2)	40.9	(+4.4)
40	34.7	35.8	(+3.2)	36.6	(+5.5)	36.7	(+5.7)
50	30.0	31.2	(+4.1)	31.2	(+4.0)	31.2	(+3.9)
60	25.1	26.2	(+4.3)	26.4	(+5.1)	26.3	(+4.6)
70	20.8	22.1	(+6.2)	22.1	(+6.0)	21.8	(+4.6)
80	16.4	17.1	(+4.7)	17.2	(+5.2)	17.0	(+4.1)
90	11.3	11.8	(+4.6)	12.0	(+6.5)	11.9	(+5.9)
100	2.3	2.7	(+17.2)	2.7	(+19.5)	2.7	(+18.6)
average	27.6	28.6	(+3.4)	28.8	(+4.0)	28.7	(+4.0)

Table 5: Improving Porter stemmer on WSJ

With the n-gram approach on WSJ the connected component algorithm generates 62,174 classes on WSJ, of average class size 1.23. The optimal partition algorithm breaks them up into 71,275 classes, with average class size 1.07. The expansion factor on the query set is 2.85 for connected component classes, and 2.28 for optimal partition classes.

With n-gram on WEST, the connected component algorithm generates 38,343 classes, average class size 1.32. The optimal partition algorithm generates 45,042 classes, average class size 1.11. The expansion factor on the query set is 3.99 for connected component classes and 2.97 for optimal partition classes.

The retrieval effectiveness of the connected component n-gram and optimal partition n-gram classes is shown in tables 6, 7 and 8. Connected component n-gram achieved about the same level of performance as KSTEM and Porter. It is worse than KSTEM and Porter on WEST, but better than KSTEM on WSJ. The optimal partition n-gram is better than KSTEM and Porter on both collections.

It is interesting to compare optimal partition n-gram with optimal partition Porter. Although they start with very different initial stemmers, they are comparable in terms of retrieval effectiveness. On WEST, optimal partition Porter is slightly better (1.0% better on natural language queries, 0.3%

better on structured queries). On WSJ, optimal partition n-gram is slightly better (0.3%).

	Precision (% change) – 34 queries				
Recall	kstem	ngramCC		ngramOptimal	
10	79.9	78.4	(−1.8)	79.4	(−0.6)
20	75.3	73.8	(−2.0)	75.8	(+0.6)
30	71.4	69.3	(−2.9)	71.2	(−0.4)
40	61.2	60.7	(−0.9)	61.6	(+0.7)
50	54.0	52.9	(−2.0)	54.9	(+1.6)
60	44.1	44.0	(−0.3)	44.7	(+1.4)
70	36.0	35.7	(−0.9)	37.1	(+2.9)
80	27.0	27.2	(+0.8)	28.6	(+5.9)
90	15.7	15.5	(−1.1)	16.6	(+5.9)
100	8.1	8.5	(+4.5)	9.0	(+10.7)
average	47.3	46.6	(−1.4)	47.9	(+1.3)

Table 6: Retrieval results of n-gram stemmers using WEST natural language queries

	Precision (% change) – 34 queries				
Recall	kstem	ngramCC		ngramOptimal	
10	80.9	80.5	(−0.5)	81.0	(+0.1)
20	75.5	74.9	(−0.8)	75.7	(+0.3)
30	72.0	71.0	(−1.4)	71.9	(−0.2)
40	63.8	63.6	(−0.3)	63.7	(−0.1)
50	57.2	56.6	(−1.0)	58.0	(+1.5)
60	49.2	49.1	(−0.4)	49.7	(+0.9)
70	41.5	40.9	(−1.5)	41.6	(+0.3)
80	30.7	31.9	(+3.8)	33.2	(+7.9)
90	19.7	19.6	(−0.2)	20.6	(+5.0)
100	9.3	9.9	(+7.1)	10.5	(+13.2)
average	50.0	49.8	(−0.4)	50.6	(+1.2)

Table 7: Retrieval results of n-gram stemmers using WEST structured queries

Figure 4 gives examples of some connected component n-gram equivalence classes. Some of them contain words not directly related. Figure 5 gives the classes produced by the optimal partition algorithm.

5.4 Scalability

Though the overhead to collect the co-occurrence information is relatively low, it would be better if we could use the co-occurrence information for part of the corpus to generate the equivalence classes

	Precision (% change) - 66 queries				
Recall	kstem	ngramCC		ngramOptimal	
10	51.7	52.9	(+2.3)	53.2	(+2.9)
20	45.0	45.6	(+1.3)	46.2	(+2.6)
30	39.2	40.3	(+2.9)	41.2	(+5.3)
40	34.7	35.8	(+3.0)	36.9	(+6.2)
50	30.0	31.4	(+4.8)	31.8	(+6.0)
60	25.1	26.5	(+5.2)	26.6	(+5.6)
70	20.8	21.5	(+3.3)	21.4	(+2.6)
80	16.4	17.0	(+3.7)	16.8	(+2.6)
90	11.3	11.8	(+4.6)	11.6	(+3.3)
100	2.3	2.6	(+12.9)	2.6	(+12.6)
average	27.6	28.5	(+3.2)	28.8	(+4.3)

Table 8: Retrieval results of n-gram stemmers on WSJ

for retrieval on the whole corpus.

Table 9 gives the retrieval effectiveness of using parts of the WSJ corpus to generate the n-gram equivalence classes. Using 10% of WSJ to generate the n-gram optimal partition classes (ngramOptimal-0.1), the retrieval performance is slightly worse than using the whole corpus (ngramOptimal). Using 50% of WSJ (ngramOptimal-0.5), the retrieval performance is as good as using all of WSJ. This means that it is possible to use a fraction of a large corpus to generate the equivalence classes without losing much performance. We need to carry out further experiments on multiple collections to decide whether the necessary amount of data is proportional to the size of the corpus, is a fixed figure, or is something else.

5.5 Portability

The results in the previous sections show that corpus analysis based on co-occurrence information helps stemming, but do not answer the questions whether stemming is corpus specific and how corpus specific it is.

To answer the questions, we use the equivalence classes generated from the co-occurrence data of one corpus to expand the queries for another one. Table 10 shows the result of applying optimal

	Precision (% change) - 66 queries				
Recall	ngramOptimal	ngramOptimal-0.1		ngramOptimal-0.5	
10	53.2	52.6	(−1.1)	53.7	(+1.0)
20	46.2	45.7	(−1.1)	46.0	(−0.4)
30	41.2	40.5	(−1.9)	41.1	(−0.3)
40	36.9	36.5	(−1.1)	37.0	(+0.3)
50	31.8	31.3	(−1.6)	31.8	(−0.1)
60	26.6	26.5	(−0.4)	26.6	(+0.1)
70	21.4	21.2	(−0.8)	21.3	(−0.5)
80	16.8	16.6	(−1.2)	16.8	(−0.3)
90	11.6	11.6	(+0.0)	11.6	(−0.5)
100	2.6	2.5	(−1.5)	2.6	(−0.1)
average	28.8	28.5	(−1.1)	28.8	(+0.0)

Table 9: Using equivalence classes for 10% and 50% of WSJ on WSJ

partition n-gram equivalence classes generated from WSJ87 (0.13 GB) co-occurrence data to WSJ91 queries. The result is better than both KSTEM and Porter, with an 4.0% improvement over KSTEM. Tables 11 and 12 show the result of applying optimal partition n-gram equivalence classes generated from WSJ co-occurrence data to WEST. The result is comparable to those of KSTEM and Porter, but clearly not as good as using the classes generated from the WEST co-occurrence data.

The results show that the stemming is somewhat corpus specific. The portability of the equivalence classes depends on the similarity of the corpora.

	Precision (% change) - 60 queries				
Recall	kstem	porter		WSJ87ngramOptimal	
10	59.8	61.6	(+2.9)	60.9	(+1.9)
20	51.5	51.6	(+0.3)	52.8	(+2.7)
30	44.5	44.7	(+0.4)	47.1	(+5.9)
40	39.1	39.2	(+0.4)	41.1	(+5.3)
50	33.0	34.2	(+3.5)	35.9	(+8.7)
60	27.8	29.1	(+4.5)	29.2	(+5.0)
70	22.6	23.9	(+6.0)	22.7	(+0.8)
80	17.7	18.4	(+4.1)	17.4	(−1.8)
90	12.2	13.2	(+8.5)	12.9	(+5.6)
100	3.8	4.4	(+15.8)	4.4	(+14.0)
average	31.2	32.0	(+2.7)	32.5	(+4.0)

Table 10: Using WSJ87 classes on WSJ91

	Precision (% change) – 34 queries				
Recall	kstem	porter		WSJngramOptimal	
10	79.9	78.2	(−2.1)	78.8	(−1.3)
20	75.3	72.8	(−3.3)	75.5	(+0.2)
30	71.4	70.1	(−1.8)	69.4	(−2.8)
40	61.2	60.2	(−1.7)	59.8	(−2.2)
50	54.0	52.9	(−2.0)	53.5	(−1.0)
60	44.1	44.5	(+0.9)	43.5	(−1.3)
70	36.0	35.5	(−1.5)	35.8	(−0.7)
80	27.0	28.7	(+6.4)	27.8	(+3.1)
90	15.7	16.5	(+4.9)	16.4	(+4.3)
100	8.1	8.8	(+7.9)	9.3	(+14.7)
average	47.3	46.8	(−1.0)	47.0	(−0.6)

Table 11: Using WSJ classes to expand WEST natural language queries

	Precision (% change) – 34 queries				
Recall	kstem	porter		WSJngramOptimal	
10	80.9	80.7	(−0.2)	80.6	(−0.4)
20	75.5	74.6	(−1.1)	77.2	(+2.2)
30	72.0	70.7	(−1.9)	71.8	(−0.2)
40	63.8	63.7	(−0.2)	64.4	(+1.0)
50	57.2	57.4	(+0.3)	58.3	(+1.9)
60	49.2	49.7	(+0.9)	49.9	(+1.3)
70	41.5	41.9	(+1.0)	40.4	(−2.6)
80	30.7	33.4	(+8.7)	31.4	(+2.2)
90	19.7	20.1	(+2.3)	20.0	(+1.8)
100	9.3	9.1	(−1.7)	9.8	(+5.8)
average	50.0	50.1	(+0.3)	50.4	(+0.8)

Table 12: Using WSJ classes to expand WEST structured queries

5.6 Efficiency

We have measured the time taken by the steps described in section 4 on WSJ on a SUN4 workstation. It takes 20 minutes CPU time to collect the unique word variants (step 1). It takes about 1 minute to construct the equivalence classes for the Porter stemmer (step 2). For the n-gram stemmer, this step is unnecessary. Collecting co-occurrence data (step 3) takes about 1 hour 10 minutes if the initial stemmer is Porter, and 1 hour 30 minutes if the initial stemmer is trigram matching. Step 4 takes less than 1 minute to run the connected component algorithm to refine the initial equivalence classes. It takes 5 additional minutes to run the optimal partition algorithm. The total time for the

	Precision (% change) – 66 queries							
Recall	delta=0.001	delta=0.0075		delta=0.01		delta=0.015		
10	53.1	53.2	(+0.2)	53.4	(+0.5)	53.2	(+0.3)	
20	46.1	46.2	(+0.1)	46.4	(+0.5)	46.3	(+0.4)	
30	41.0	41.2	(+0.7)	41.3	(+0.8)	41.0	(+0.2)	
40	36.7	36.9	(+0.5)	36.8	(+0.3)	36.2	(−1.2)	
50	31.6	31.8	(+0.7)	31.7	(+0.3)	31.0	(−1.9)	
60	26.6	26.6	(−0.1)	26.5	(−0.2)	26.1	(−1.8)	
70	21.5	21.4	(−0.6)	21.2	(−1.4)	21.0	(−2.2)	
80	16.7	16.8	(+0.4)	16.7	(−0.1)	16.7	(−0.1)	
90	11.6	11.6	(+0.4)	11.6	(+0.3)	11.8	(+1.6)	
100	2.6	2.6	(−0.6)	2.6	(+1.3)	2.5	(−2.4)	
average	28.7	28.8	(+0.3)	28.8	(+0.3)	28.6	(−0.5)	

Table 13: Effect of δ on ngram stemmers on WSJ

most expensive configuration (n-gram optimal partition) of the steps 1–4 is about two hours.

In a production environment, step 1 can be omitted because the indexing phase can give us the unique word variants. Together with a more sophisticated implementation of step 3, we are confident that the total time for steps 1–4 can be reduced to 1 hour on WSJ. A speed of 500 MB per hour is faster than the indexing speed of typical IR systems, so applying corpus analysis to stemming is practical.

With query-based stemming (query expansion using equivalence classes) retrieval time varies with the stemmer used. We found that there is a direct correlation between the expansion factor and retrieval time. For example, INQUERY takes 2 hours 27 minutes CPU time to finish the 66 WSJ queries expanded by Porter stemmer and 1 hour 23 minutes CPU time to finish the same queries expanded by the n-gram optimal partition stemmer. The expansion factors of the two stemmers on WSJ are 4.5 and 2.28. Because the technique in this work significantly reduces the expansion factor, it can result in much faster retrieval that is especially desirable for interactive applications.

Query-based stemming is, however, less efficient than stemming at document indexing time. It takes only 42 minutes CPU time to finish the 66 WSJ queries on a pre-stemmed WSJ database. We think the efficiency gap is largely due to the fact that the current implementation of INQUERY is not

well tuned for query-based stemming. On a database stemmed at indexing time, each equivalence class needs one disk I/O to get the inverted list. With query based stemming, each equivalence class needs several disk I/O's to read the inverted lists for all class members and merge them. Using the same stemmer, the disk transfer time is about the same. So the extra overhead required by query based stemming is the extra disk seeks and the merge of the inverted lists. Using the Porter stemmer on WSJ, query based stemming requires 130 extra disk seeks per query on average, which take about 2 seconds on a workstation. The overhead to merge the inverted lists is modest on a modern computer system. When properly implemented, query-based stemming in combination with the corpus analysis technique in this work may be even more efficient than stemming at document indexing time using traditional stemmers because corpus analysis results in much smaller equivalence classes and shorter inverted lists for the query words.

6 Spanish Experiments

It is natural to generalize corpus analysis and n-gram matching to other languages. We have tried these ideas on Spanish and the preliminary results are encouraging. To simplify the software design and the experiments, accents are removed from the Spanish characters and upper cased characters are lowercased in the ISM corpus. The retrieval performances of the n-gram connected component and n-gram optimal partition stemmers on the ISM corpus are shown in table 14. The parameter values (window size, *em* threshold and δ) are the same as in the English experiments. The baseline stemmer for the Spanish experiment is a Porter style stemmer designed at the Center for Intelligent Information Retrieval at University of Massachusetts at Amherst and used by UMASS in the Spanish experiments of TREC conferences [Broglio et al. 1995]. As in the English case, the retrieval performance of the n-gram stemmers is comparable with that of the baseline stemmer. But interestingly, unlike in the English experiments, the optimal partition stemmer is slightly worse than

the connected component stemmer.

	Precision (% change) – 25 queries				
Recall	baseLineStemmer	ngramCC		ngramOptimal	
10	48.9	49.6	(+1.4)	47.8	(−2.3)
20	39.3	39.6	(+0.8)	39.6	(+0.9)
30	31.8	33.7	(+6.0)	32.6	(+2.3)
40	26.4	26.8	(+1.7)	26.1	(−1.2)
50	21.6	21.9	(+1.3)	19.9	(−8.0)
60	16.4	16.9	(+2.7)	16.6	(+1.2)
70	12.6	12.6	(+0.2)	12.6	(+0.4)
80	9.1	9.2	(+1.1)	8.9	(−1.8)
90	4.9	5.0	(+1.4)	5.2	(+4.7)
100	0.7	0.5	(−32.5)	0.6	(−18.1)
average	21.2	21.6	(+1.9)	21.0	(−0.9)

Table 14: Retrieval Performances on ISM

The ISM corpus has 221,095 unique word forms and the expansion factor of the baseline stemmer is 20.4. The expansion factors of the n-gram connected component and n-gram optimal partition stemmers on ISM are 8.1 and 3.4. Because Spanish is very rich in morphology, these figures are not surprising.

Recall that the parameters of the class refinement algorithms control the sizes of the resulted equivalence classes. Because Spanish is richer morphologically than English, we might expect that the ideal parameter values for English and Spanish are different. Using the same parameter values as in the English experiments may prevent many good conflations for Spanish. One of the reasons why the optimal partition algorithm is slightly worse than the connected component algorithm may be that it produces equivalence classes that are too small for Spanish. The other possible reason is that our current implementation of the optimal partition algorithm can not handle input equivalence classes of more than 12 members and we have to break them up into pieces before running the algorithm. This may break up good conflations. Further work needs to be done to decide what are the appropriate parameter values for Spanish. We are aslo looking at inexpensive approximations of the optimal partition algorithm to handle large input classes.

7 Conclusion

The results show that corpus-based analysis of word variants can be used to enhance the performance of stemming algorithms. Experiments with crude equivalence classes generated by trigram matching demonstrate the validity of the basic hypothesis that related word variants co-occur in limited text windows.

We have begun to work on two issues that come out of this research. One is improving the efficiency of query-based stemming. One solution would be to index both stems and word variants, and only use the word variants when specified by the user. This of course increases the index overhead substantially, which is why we are also looking at increasing the efficiency of query processing with many word variants.

The second issue is stemming in non-English languages. A statistical approach to stemming is a potentially effective way of improving a retrieval system for a given language without needing a language expert to produce a stemmer. Initial experiments with Spanish using the trigram approach have been encouraging. The problem in languages with more complex morphology, such as Arabic, is generating the initial equivalence classes. It may be that simply expanding the query using all related words rather than just word variants will accomplish the same effect.

Acknowledgments

Thanks to James Callan for his helpful comments on an early draft of this paper. Thanks to Lisa Ballesteros for her help in the Spanish experiments. Thanks to the WEST Publishing Company for providing the WEST collection of legal texts.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623 and

NRaD Contract Number N66001-94-D-6054. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

[Broglio et al. 1994] Broglio, J., Callan, J. P., AND Croft, W. (1994). An overview of the INQUERY system as used for the TIPSTER project. In *Proceedings of the TIPSTER Workshop*, pp. 47–67. Morgan Kaufmann.

[Broglio et al. 1995] Broglio, J., Callan, J. P., Croft, W. B., AND Nachbar, D. W. (1995). Document Retrieval and Routing Using the INQUERY System. In Harman, D., editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pp. 22–29. NIST Special Publication 500-225.

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

[Croft and Xu 1995] Croft, W. B., AND Xu, J. (1995). Corpus-specific stemming using word form co-occurrence. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, pp. 147–159.

[Harman 1991] Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science*, 42(1),7–15.

[Harman 1995] Harman, D. (1995). Overview of the Third Text REtrieval Conference (TREC-3). In Harman, D., editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pp. 1–20. NIST Special Publication 500-225.

[Hull 1996] Hull, D. A. (1996). Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1),70–84.

[Jing and Croft 1994] Jing, Y., AND Croft, W. (1994). An association thesaurus for information retrieval. In *Proceedings of RIAO 94*, pp. 146–160.

[Krovetz 1993] Krovetz, R. (1993). Viewing morphology as an inference process. In *Proceedings of the 16^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 191–202.

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

[Salton 1989] Salton, G. (1989). *Automatic Text Processing.* Addison-Wesley, Reading, MA.

[Turtle 1994] Turtle, H. (1994). Natural language vs. Boolean query evaluation: A comparison of retrieval performance. In *Proceedings of the 17^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 212–220.

[van Rijsbergen 1979] van Rijsbergen, C. (1979). *Information Retrieval.* Butterworths, second edition.

[Voorhees 1994] Voorhees, E. (1994). Query expansion using lexical-semantic relations. In *Proceedings of the 17^{th} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 61–69.

```
marketed marketer markets marketing mark marker marked marks
 -marketers markers market
uniformity uniformly uniformed uniforms uniform
generated generating generation generator generators
 -generations generate generates generalize
```

Figure 4: examples of n-gram connected component classes on WEST

```
marked
marketer marketers
marker markers
marketed markets marketing mark marks market
uniformed uniforms
uniformity uniformly uniform
generator generators
generations
generated generating generation generate
generates generalize
```

Figure 5: examples of n-gram optimal partition classes on WEST