Details on Stemming in the Language Modeling Framework

James Allan and Giridhar Kumaran
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA
{allan,giridhar}@cs.umass.edu

ABSTRACT

We incorporate stemming into the language modeling framework. The work is suggested by the notion that stemming increases the numbers of word occurrences used to estimate the probability of a word (by including the members of its stem class). As such, stemming can be viewed as a type of smoothing of probability estimates. We show that such a view of stemming leads to a simple incorporation of ideas from corpus-based stemming. We also present two generative models of stemming. The first generates terms and then variant stems. The second generates stem classes and then a member. All models are evaluated empirically, though there is little difference between the various forms of stemming.

1. INTRODUCTION

Stemming is the process of collapsing words into their morphological root. For example, the terms *addicted*, *addicting*, *addiction*, *addictions*, *addictive*, and *addicts* might be conflated to their stem, *addict*. In information retrieval (IR) systems, stemming serves to aid one or both of:

- efficiency—limiting the number of unique words reduces the size of an IR system's dictionary, can improve compression rates [11], etc.
- effectiveness—in theory, stemming improves a system's recall of relevant material. Documents that contain morphological variants of query words have a good chance of also being relevant.

Over the years, numerous studies and countless classroom projects have explored the effectiveness issues of stemming from almost any angle imaginable: should stemming be used at all, how can stemmers be improved, what advantages do different stemmers provide, how can stemming be done in new languages, and so on.

In this study, we step back from that style of experiment somewhat and explore the question of what precisely stemming accomplishes. We are motivated by the observation[1] that stemming can be viewed as a form of smoothing, as a way of improving statistical estimates. If the observation is correct, then it may make sense to incorporate stemming directly into a language model rather than treating it as an external process to be ignored or used as a pre-processing step. Further, it may be that viewing stemming in this light will illuminate some of its properties or suggest alternate ways that stemming could be used.

In this study, we tackle that problem, first by converting classical stemming into a language modeling framework and then showing that used that way, it really does look like other types of smoothing. This view of stemming will suggest an obvious extension that begs being merged with ideas from corpus-based stemming.

Once the idea of stemming is embedded in the language modeling framework, other ways that it can be included start suggesting themselves. We will briefly touch on two ways of viewing stemming as a generative process rather than merely a technique to improve statistical estimates.

The focus of this work is on a different way to view stemming. However, we will also report on a series of experiments that evaluate the effectiveness of different models along the way. The experiments will show modest, but rarely statistically significant, improvements in comparison to the simplest form of stemming. All forms of stemming will result in better accuracy than omitting stemming.

The following section reviews related work in stemming. In Section 3 we briefly review some ideas from language modeling, probability estimation, and smoothing that are central to this paper. We then describe in Section 4 the experimental setup in which we carried out our empirical validations. The core of the paper starts in Section 5 where we incorporate stemming into the language modeling framework and then, in Section 6, briefly flirt with the idea of partial stemming suggested by doing so. We then show in Section 7 how stemming can be treated as a form of smoothing. That leads to the obvious idea discussed in Section 8 of allowing different words to contribute differently to the smoothing of a word's probability. Then in Section 9 we switch gears to develop and evaluate two generative models of stemming, one that is similar in spirit to a translation model. We summarize our findings in Section 10.

2. PREVIOUS RESEARCH

[1]An observation that was first expressed to us by Jay Ponte at a May/June 2001 workshop on language modeling for information retrieval.

We discuss work exploring the question of whether or not stemming provides more effective results. We also outline ideas behind corpus-based stemming, a statistical approach to generating stem classes.

2.1 Stemming effectiveness

There have been several in-depth studies investigating the effectiveness of stemming on document retrieval. Harmans evaluation of the performance of three different stemming algorithms, namely the S-stemmer, Lovins stemmer, and Porter stemmer, concluded that stemming failed to deliver any improvement in performance [3]. While quite a few queries did benefit from stemming, an equal number were in fact impaired. The net improvement was thus inconsequential. Harman suggested possible improvements in stemming, one of which had to do with treating different words differently—the formal model of stemming presented in this study will inspire precisely that idea (though we will not find a substantial improvement in effectiveness).

Krovetz experimented with an inflectional stemmer, a derivational stemmer, and a revised Porter stemmer that used a dictionary to check if a resulting stem exists [5]. In contrast to Harman's earlier study, all the stemmers resulted in improved performance, with the derivational stemmer performing the best. Krovetz also found that stemming was better suited for short documents and queries.

Hull questioned the use of traditional precision and recall measures for evaluating experiments involving stemming algorithms [4]. Traditional measures do not reflect the effect of length of the queries, length of the documents, nature of the corpus, the number of relevant documents for a query, and so on. Hull concluded that stemming is almost always beneficial expect for long queries at low recall levels.

Stemmers that are tailor-made for languages that are highly inflectional in character definitely influence retrieval performance [9, 6].

We know of no work that viewed stemming as a formal process within a probabilistic framework. Typically stemming has been included in information retrieval systems in an ad-hoc manner, justified for reasons of efficiency or empirical evaluations. Despite the evidence supporting and wide adoption of stemming in research systems, modern Web search engines usually do not stem their collections. For commercial reasons, they do not support this decision, but it appears that for queries that are not recall oriented, their collections are so large that missing Web pages because of morphological variance is not a concern.

2.2 Corpus-based stemming

Conventional stemming algorithms employ morphologically-derived rules to conflate word variants to their roots. Although such approaches are generally effective, they do not consider the influence of the language in specific corpora. As an example, *ford* and *fordable* are related in documents on nature but are unrelated in documents on automobiles. In other words, a conflation that is useful for one corpus might be harmful to another. Similarly, blind application of rules may result in wrong conflations such as *homing* with *homely*.

One effort to address those problems is called corpus-based stemming [12]. The hypothesis of that work is that word forms that should be conflated will co-occur in documents from the corpus. It starts with a set of rough preliminary stem classes created by another stemmer, perhaps Porter or something that conflates all words starting with the same three letters. It then uses co-occurrence analysis of the words in the preliminary class to find ones that do not appear to belong together. Corpus-based stemming was found to provide moderate improvement over existing rule-based stemmers.

The co-occurrence analysis is based on *em*, a variation of *EMIM* (expected mutual information measure). *EMIM* is widely used to measure the significance of word associations.

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

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

The *EMIM* measure does not work well directly because it is not normalized over the number of occurrences of a and b and unfairly favors high frequency words. The *em* metric is defined as:

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

where $En(a,b)$, the expected number of co-occurrences of a and b, is kn_an_b, where k is a constant based on the corpus and window size. k is estimated from a sample of 5000 randomly chosen word pairs:

$$k = \frac{\sum n_{ab}}{\sum n_a n_b}$$

$En(a,b)$ plays an important role as it reduces the scores of two words that co-occur simply by chance.

3. LANGUAGE MODELING

In this work, we focus on the query-likelihood variant of statistical language modeling (as opposed to, for example, document likelihood models). The techniques model the query generation process as a random sampling of the probabilistic model of a topic as suggested by a document [8]. That is, given a query $Q = q_1q_2q_3\ldots q_n$, and a document $D = d_1d_2d_3\ldots d_n$, we wish to estimate $P(Q|D)$, the probability that the query would be generated by the document. We make the traditional assumption of term independence to get:

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

Strictly speaking, the value D in that equation represents a model (probability distribution), the only information about which we have is that D comes from the model. A good deal of research in the field investigates how to reliably estimate a model given such sparse information [7, 14]

To start with, however, almost every language modeling approach uses a maximum likelihood estimate:

$$P_{ML}(w|D) = \frac{c(w;D)}{\sum_{i=1}^{n} c(w_i;D)}$$

where $c(w;D)$ represents the number of times that term w occurs in document D.

Table 1: Statistics of the corpora used for empirical validation of the models in this study.

Dataset	NumDocs	Size	QueryIDs
AP89	84,678	254Mb	1-50 routing
AP90	78,321	237Mb	101-150 ad-hoc
FBIS	130,471	470Mb	1-243 routing
TREC5	524,929	1.9Gb	251-300 ad-hoc
TREC8	528,155	2.05Gb	401-450 ad-hoc

Unfortunately, a problem arises when a term does not appear in the document D. Irrespective of the counts of the other terms in the query, $P(Q|D)$ will end up being zero because one of the $P(q_i|D)$ will be estimated as zero. To avoid this problem, smoothing is employed to assign a non-zero probability to the unseen words—and as an added benefit, to improve the estimation of word probabilities in general (because D is only one sample and may not be a great sample for every word).

A wide range of smoothing techniques have been adapted for information retrieval [8, 15, 16]. Broadly speaking, these methods discount the probabilities of the terms observed, and assign some extra probability mass to the unseen terms according to some fallback model. In information retrieval, a technique called Jelinek Mercer smoothing [15] is most often employed. This simple mixture model involves the interpolation of the maximum likelihood model with an estimate derived from a large collection of documents (called a "background" or a "general english" model). Thus:

$$P(w|D) = (1-\lambda)P_{\text{ml}}(w|D) + \lambda P_{\text{ml}}(w|\text{collection}) \quad (1)$$

where the coefficient λ is used to control the influence of each estimate. The value of the coefficient can be set via training data, heuristically, or based on a more elaborate theory of modeling language [16].

Throughout this study, we will *always* smooth our probability estimates using the entire evaluation set as a background collection (e.g., queries run against AP89 will use the entire AP89 collection to form a background model). Whatever means we use to estimate the probability from just the document (e.g., maximum likelihood) we will also employ to estimate the background probability. Then we will interpolate those two estimates based on the value of λ.

4. EXPERIMENTAL SETUP

In the remainder of this study, we will investigate several views of stemming within the language modeling framework. We will run experiments showing the impact—usually negligible—of the different choices by using five different collections taken from the TREC corpora as listed in Table 1.

The FBIS documents are from TREC volume 5, and the queries are the 50 routing queries from TREC 5's routing task. TREC5 is made up of TREC volumes 2 and 4, the set of documents used for the TREC-5 ad-hoc retrieval track. TREC8 consists of TREC volumes 4 and 5 with the congressional record removed, the set used for the TREC-8 ad-hoc track.

The queries used were either the ad-hoc or routing queries prescribed for each corpus. The title and description portions from each TREC topic were used to formulate the queries for our experiments.

Table 2: Impact of incorporating stemming into the unigram language modeling framework.

Dataset	Unstemmed	Stemmed	Improvement
AP89	0.2491	0.2725	+8.58%
AP90	0.2337	0.2632	+12.6%
FBIS	0.1010	0.1460	+44.5%
TREC5	0.1356	0.1565	+15.4%
TREC8	0.2167	0.2626	+21.1%

We used version 1.1 of the open source Lemur system[2] for all of our experiments, with substantial modification to support a variety of stemming methods. We used the 418 stopwords included in the stop list used by InQuery [2]. We ran Lemur with its normal stemming disabled so that it indexed all forms of words; we will reintroduce stemming in several alternate ways below. When we use the Porter stemming algorithm [10] to find stem classes, we are using the implementation provided as part of Lemur.

5. STEMMING IN LANGUAGE MODELS

Given a collection of documents that has been indexed without any stemming, the normal model described above represents using a language modeling approach without stemming:

$$P_{\text{unstem}}(w|D) = P_{\text{ml}}(w|D) = \frac{c(w;D)}{\sum_{i=1}^{n} c(w_i;D)} \quad (2)$$

(smoothed with a background corpus according to λ).

The simplest way to incorporate stemming into the language model would be to stem the collection before indexing—i.e., to index documents consisting of word stems. Short of doing that, we can simulate a stemmed collection by calculating the probability of a word using all words in its stem class. For example, to estimate the probability that *addicted* occurs in a model, we could count not just occurrences of *addicted* as in Equation 2, but also occurrences of *addicting*, *addiction*, *addictions*, and so on. This leads to:

$$\begin{aligned} P_{\text{stem}}(w|D) &= \frac{\sum_{w_j \in E(w)} c(w_j;D)}{\sum_{i=1}^{n} c(w_i;D)} \\ &= \sum_{w_i \in E(w)} P_{\text{unstem}}(w_i|D) \end{aligned} \quad (3)$$

where $E(w)$ represents the equivalence or stem class of w—that is, all words w_i that have the same stem as w (obviously, $w \in E(w)$).

Calculating the query likelihood based on Equation 3 provides exactly the same results as if Equation 2 were used on a stemmed version of the collection. Similarly, if Equation 3 is used with stem classes of size one (i.e., no stemming), it reduces to Equation 2.

We used Equations 2 and 3 to provide baseline runs for all five of our test collections. All runs were done using unstemmed versions of the collections; "stemming" was done at query time by using Equation 3 to calculate the probabilities. Table 2 summarizes the mean average precision numbers for all collections, with and without stemming.

It is clear that stemming provides a noticeable improvement in effectiveness when used with a very simple language

[2] `http://www.cs.cmu.edu/~lemur`

model. Figure 1 shows the recall precision tradeoff graphs for the AP89 collection, where gain from stemming is the least. The tradeoff graphs are similar for the other collections.

Table 3 shows results for all of the approaches that will be discussed in this paper. Perhaps the most surprising result is that although there are huge differences in average precision when stemming is applied, the difference is rarely statistically significant.

6. PARTIAL STEMMING

It is common in language modeling to combine multiple estimates to provide a better estimate. The interpolation of Jelinek Mercer smoothing (Equation 1) is an example of combining an estimate from a small sample of text with an estimate from a larger background corpus.

Equations 2 and 3 are both ways of estimating the probability of occurrence of a word (though the latter actually estimates the probability of the word class), which immediately suggests the possibility of combining those estimates by interpolation:

$$P_{\text{partial}}(w|D) = \alpha P_{\text{unstem}}(w|D) + (1-\alpha)P_{\text{stem}}(w|D)$$

In a sense, the probability is a combination of the estimate on a small sample (a single word's occurrences) combined with that of a larger sample (all words in the stem class). This combination allows a system to progress smoothly between stemming ($\alpha = 0$) and no stemming ($\alpha = 1$), and begs the question of what it means for α to lie in the middle of the range.

We tried some experiments to see if a globally optimal value of α could be found that was somewhere between the extremes. For any given set of data and queries, we could easily find values that improved upon the endpoints. For example

- For the AP89 collection, setting $\lambda = 0.6, \alpha = 0.8$ resulted in an average precision of 0.2811, meaning that 80% stemming resulted in a 3% improvement over 100% stemming.
- For the TREC5 collection, optimal performance appeared with $\lambda = 0.5, \alpha = 0.3$, resulting in a modest improvement of 1% to 0.1586.

We were unable to find values of α and λ that would work on a range of collections, and were unsuccessful in devising a strategy for estimating the parameters.

Another way to treat α, though, is to have it vary on a query-by-query (or even word-by-word) basis. That is, if it were possible to decide for a query (or a word) whether or not stemming was appropriate, perhaps performance could be improved. We "cheated" in Figure 2 to find out how good a system might be able to do if it could decide whether α should be 0 or 1, by selecting the value that did this best. The graph makes it clear that being able to decide whether to stem would be a win. Unfortunately, we were unable to establish criteria for selecting α: Harman's conclusions [3] may be irrefutable.

7. STEMMING AS SMOOTHING

The previous section uses smoothing (interpolation of two estimates) to incorporate stemming into the language mod-

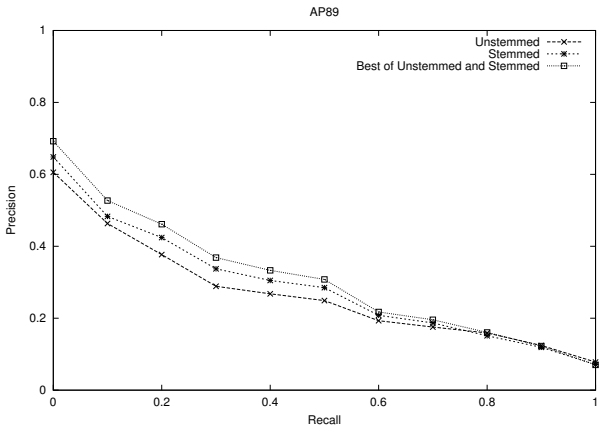

Figure 1: Unstemmed and stemmed retrieval recall–precision graphs for the AP89 collection.

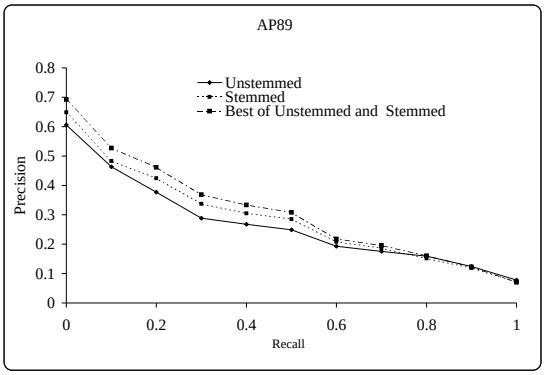

Figure 2: For the AP89 collection, shows performance for stemming, not stemming, and a system that selects one or the other on a per-query basis. These numbers were obtained by knowing the answer.

Dataset	Metric	Unstemmed	Stemmed	Xu& Croft	Co-occur	1,000 Stemmers	Term Gen	Class Gen
AP89	AvgPrec	0.2491	0.2725	0.2819*	0.2799	0.2799	0.2779*	0.2726
	P@20	0.2389	0.2378	0.2478*	0.2411	0.2441	0.2389	0.2378
AP90	AvgPrec	0.2337	0.2632	0.2660	0.2650	0.2650	0.2650	0.2656
	P@20	0.3410	0.3460	0.3530	0.3510	0.3510	0.3570	0.3600
FBIS	AvgPrec	0.1010	0.1460	0.1403*	-	-	0.1453	0.1460
	P@20	0.2100	0.2222	0.2222	-	-	0.2200	0.2222
TREC5	AvgPrec	0.1356	0.1565	-	-	-	-	-
	P@20	0.2340	0.2570*	-	-	-	-	-
TREC8	AvgPrec	0.2167	0.2626*	0.2639	-	-	0.2636	0.2626
	P@20	0.3750	0.4040	0.4090	-	-	0.4050	0.4040

Table 3: Performance of algorithms discussed in paper, measured in terms of average precision and precision at 20 documents retrieved. An asterisk indicates results that are significantly different ($P < 0.05$) from the stemmed results, except in the stemmed column where it is compared to the unstemmed results. Missing values represent experiments that were not run.

eling framework. However, it does not really treat stemming as actual smoothing.

The format of Equation 3 is reminiscent of the formula for smoothing (Equation 1). The similarity is clearer if the former is rewritten:

$$P(w|D) = P_{\text{ml}}(w|D) + \sum_{\substack{w_i \in E(w) \\ w_i \neq w}} P_{\text{ml}}(w_i|D)$$

The conversion to smoothing is accomplished by adding an interpolation parameter β:

$$P(w|D) = \beta P_{\text{ml}}(w|D) + (1-\beta) \sum_{\substack{w_i \in E(w) \\ w_i \neq w}} P_{\text{ml}}(w_i|D)$$

When $\beta = 1$, the equation reduces to unstemmed retrieval. When $\beta = 0.5$ all words in the stem class (including the original word w) are treated equally. In that case, even though the probabilities would be different, the ranking of the system would be identical to that of P_{stem}. We do not perform any experiments on this approach since the ranking would not change.

We have now represented stemming as a true smoothing process. The probability of a word is calculated as an interpolation of its own probability and the probability of all other words in its stem class.

8. ADHOC MIXTURE MODELS

We generalize the view of stemming in the previous section as follows:

$$P_{\text{mix}}(w|D) = \frac{1}{\sum_j f(w_j, w)} \sum_{w_i \in E(w)} f(w_i, w) P_{\text{ml}}(w_i|D)$$

where $f(w_i, w)$ is a function that indicates how much significance term w_i has in calculating the probability for word w. This results in an interpolation over the probabilities for all of the words in the stem class. This general form admits a range of possibilities depending on how $f()$ is defined.

If $f(w_i, w) = 1$ when $w_i = w$ and 0 otherwise, then $P_{\text{mix}} = P_{\text{unstem}}$, because in that case only the actual word form will be used to estimate its probability.

On the other hand, if $f(w_i, w) = 1/|E(w)|$, then $P_{\text{mix}} \propto P_{\text{stem}}$. In fact, $P_{\text{mix}} = P_{\text{stem}}/|E(w)|$ which means that they differ only by a constant. The resulting probabilities will be different, but the ranking of documents in an information retrieval system will not be affected. This is equivalent to the formula at the end of Section 7, where $\beta = 1/|E(w)|$.

Note that although the rankings given by P_{mix} and P_{stem} are identical, the probabilities are differently motivated. In the case of P_{stem}, the probability of a word is calculated by treating the entire stem class as if it were that word. We are really calculating the probability of the stem class. For P_{mix}, on the other hand, the probability of a word is the average of a set of estimates, one from each word in the stem class. The probability that a particular word would be generated depends not only on its probability in the model, but also on the probability of other, related, words.

All of the ways of estimating $P(w|D)$ that we have discussed so far treat every word in a stem class identically or privilege only the original word w. However, it may make more sense to give priority to terms that are meaningfully related to the original query term. For example consider an equivalence class generated by the Porter stemmer: *append*, *appended*, *appendicitis*, and *appending*. In estimating the probability of the term *append* using a mixture model, the influence of the term *appendicitis* would ideally be given very little weight, and certainly much less weight than would *appended*.

To address this idea, we now consider variations of P_{mix} that use different forms of $f()$. We explore two possibilities. In the first, we use the co-occurrence analysis of corpus-based stemming to determine the value of $f()$. In the second, we imagine large numbers of stemmers and let $f()$ represent the chance that words would be put together by those stemmers.

8.1 Corpus-based stemming

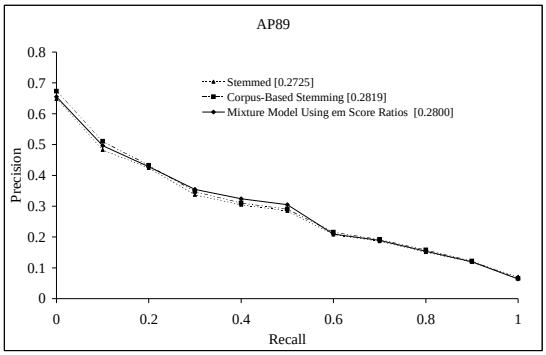

Figure 3: Recall/precision curve obtained for AP89 collection using the co-occurrence based mixture model. The curve is compared with that obtained by stemmed retrieval, as well as corpus-based stemmed retrieval.

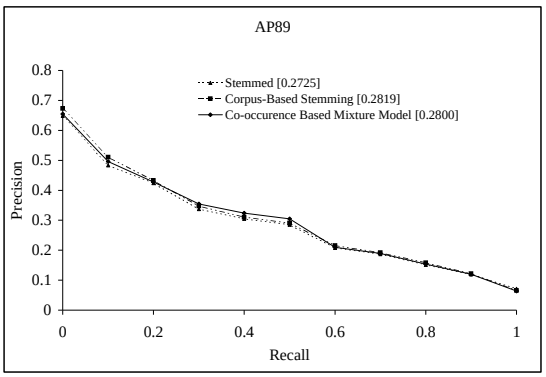

Figure 4: Recall/precision curve obtained for AP89 collection using the co-occurrence based mixture model that works by utilizing 1,000 different stemming algorithms. The curve is compared with that obtained by simple stemmed retrieval and corpus-based stemming.

In corpus-based stemming (see Section 2.2), *em* scores are calculated for every pair of terms in the preliminary class. Then a threshold is determined, scores below that are discarded, and the class is partitioned into its resulting connected components. The hope is that non-co-occurring words are not actually morphologically related (in this corpus) and should be in different classes.

In our case, we are not interested in selecting a threshold. However, we are interested in a measure that predicts whether or not two words are likely to be morphological variants of the same word. That is, a measure that will give a high score to *ford* and *fordable* in a nature corpus, but a low score in an automobile corpus. To that end, we define P_{em} to be P_{mix} with $f(w_i, w) = em(w, w_i)$, so:

$$P_{\text{em}}(w|D) = \frac{1}{\sum_j em(w_j, w)} \sum_{w_i \in E(w)} em(w_i, w) P_{\text{ml}}(w_i|D)$$

(As always, we also smooth this estimate using the background collection as indicated by the value of λ.)

Figure 3 compares this approach to the stemmed baseline and to the corpus-based stemming approach of Xu and Croft; results are also included in Table 3 for comparison. Although the heuristic techniques used for corpus-based stemming outperform the co-occurrence model based on *em*, they both do better than simple stemming, and the differences are small.

8.2 Many stemmers

Here we consider an alternate method of determining the likelihood that two words are in the same stem class. We will implement it using the corpus-based stemming approach just outlined, but that is a convenience and not a necessity.

Suppose that we had access to 1,000 different stemming algorithms and that we had applied them all to our corpus.[3]

[3]This idea was suggested by Jay Ponte.

We could measure the probability that words w_i and w_j are in the same stem class by just counting the proportion of those 1,000 stemmers that conflated them. If $n(w, w_i)$ is the number of stemmers that put a and b into the same stem class, we could then set $f(w, w_i) = n(w, w_i)/1000$ to get,

$$P_{1000}(w|D) = \frac{\sum_{w_i \in E(w)} n(w, w_i) P_{\text{ml}}(w_i|D)}{\sum_j n(w_j, w)}$$

We create our 1,000 different stemmers by varying the threshold on the corpus-based stemming algorithm described above.[4] So for the first stemmer, the original equivalence classes are used. For the second, very weak connections are broken. By the 1000th stemmer, all connections have broken and every word is in its own stem class (i.e., no stemming). Word pairs that are very tightly related in the corpus will have a high *em* score and so will be together for most threshold values, meaning a high probability of being in the same class.

The results of this approach are summarized in Table 3. We also illustrate the differences with two recall precision graphs. The first, Figure 4 compares simple stemming, corpus-based stemming, and the 1,000-stemmers model. The new model falls between the other two. The difference between the models appears to lie mostly in the top-ranked documents, so Figure 5 shows the precision across the those documents. Interestingly, after about 30 documents retrieved, there seems to be no difference at all in the different stemmers.

9. GENERATIVE MODELS

All of the models above provided ad-hoc techniques for mixing the probabilities of words in a stem class. In this

[4]We cannot imagine creating even 100 truly distinct algorithms.

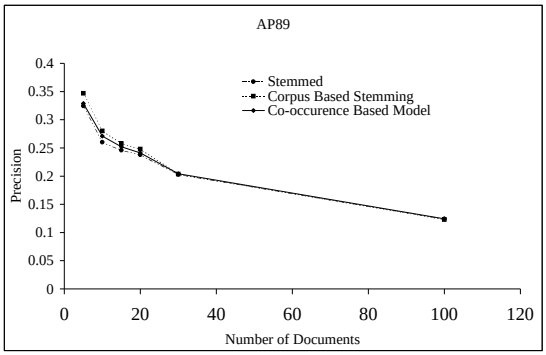

Figure 5: Precision for top-ranked documents for the AP89 collection using the co-occurrence based mixture model that works by utilizing 1,000 different stemming algorithms. The curve is compared with that obtained by simple stemmed retrieval and corpus-based stemming.

section we consider two alternate approaches to modeling stemming, both based on a generative model.

For our first approach, consider this process for generating a (query) word given a document model. First we generate a random word w_i from the vocabulary. Then we select a (query) word w (which might be w_i itself) that is a morphological variant of w_i and output w. This term generative model can be represented as:

$$P_{\text{tgen}}(w|D) = \sum_{w_i} P(w|w_i)P(w_i|D)$$

where $P(w_i|D)$ is the probability of selecting the word w_i from the model and $P(w|w_i)$ is the probability of selecting w as the morphological variant of w_i to output. We estimate the latter probability by the proportion of windows containing w_i that also contain w:

$$P(w|w_i) = n(w, w_i)/n(w_i)$$

This term generation approach is the same as the translation models that are common in language modeling approaches to cross-language retrieval [13] or to within-language retrieval [1]. The difference is that any "translations" are done within the same language and are restricted to words within the same stem class. (In theory, the summation is over all words in the vocabulary, but any words w outside the stem class have $P(w|w_i) = 0$.)

These effectiveness of this term generation technique is included in Table 3. It is comparable in effectiveness to the other approaches.

Another type of generative model is based on the intuition that a writer might think of a concept and then choose the appropriate variant of that concept depending on the situation. Specifically, the model first generates a stem class c and then selects from that class one of its words w to output—by earlier notation, $c = E(w)$, but we choose the class and then the word. Formally,

$$P_{\text{cgen}}(w|D) = \sum_{c} P(w|c)P(c|D)$$

where $P(c|D)$ represents the probability of choosing a particular class and $P(w|c)$ is the chance that the word w would be chosen. We estimate the latter as the collection frequency of the term w divided by the sum of the collection frequencies of all the terms in the equivalence class c(i.e., common words in the class are more likely to be generated). $P(c|D)$ can be calculated by counting the number of words in the stem class c that occur in D and dividing the the length of D (i.e., the proportion of D's words that are in stem class c).

The effectiveness of the class generative approach is given in Table 3. The results are comparable to the other models.

10. CONCLUSIONS

We have examined word stemming from a language modeling perspective. We showed how stemming could be brought into the language modeling framework and how that suggested a notion of partial stemming. We were able to find parameter values that made partial stemming outperform both stemming and not-stemming, but we did not find a way to estimate those parameters reliably.

We then developed a model of stemming as the interpolation of probability estimates from each word within a stem class. That led to the obvious generalization that treated different words in the class differently, allowing different styles of stemming depending on the corpus or the context. We showed how this idea could be integrated with corpus-based stemming in two different ways.

Finally we presented two generative models of stemming, one that is an analogue to a translation model and another that is more unique to the stemming process.

In all cases we provided empirical results showing how well the various approaches to stemming worked on a variety of collections. On these corpora with a simple unigram language modeling approach, stemming provide definite improvements over not stemming (Section 5). However, the several alternative views of stemming rarely improved on the effectiveness of stemming and, when they did, did so only by a small amount.

The work in this paper has not resulted in a vast improvement in stemming capabilities. We hope that by treating stemming in a range of probabilistic ways, some aspects may be better illuminated. For example, this approach suggests that we might be able to make use of better estimates of the probability of two words being in the same class. It also indicates that having more information about the probability of a particular morphological variant being chosen (which words are more common) can be readily incorporated.

It is our hope that better probability estimates may actually improve the effectiveness of retrieval systems, and that additional work will suggest alternate and more powerful models for representing the stemming process. We intend to move forward in those directions. We are also pondering the concept of creating 1,000 different stemmers.

11. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by SPAWARSYSCEN-

SD grant numbers N66001-99-1-8912 and N66001-02-1-8903. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

12. REFERENCES

[1] A. Berger and J. D. Lafferty. Information retrieval as statistical translation. In *Proceedings of ACM SIGIR1999*, pages 222–229, 1999.

[2] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

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

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

[5] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of ACM SIGIR93*, pages 61–81, 1998.

[6] L. Larkey, L. Ballesteros, and M. Connell. Improving stemming for arabic information retrieval: Light stemming and co-occurrence analysis. In *Proceedings of ACM SIGIR*, pages 269–274, 2002.

[7] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Proceedings of ACM SIGIR2001*, pages 120–127, 2001.

[8] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of ACMSIGIR98*, pages 275–281, 1998.

[9] M. Popovic and P. Willett. The effectiveness of stemming for natural language access to slovene textual data. *JASIS*, 43(5):191–203, 1993.

[10] M. F. Porter. An algorithm for suffix stripping. *Program (Automated Library and Information Systems)*, 14(3):130–137, 1980.

[11] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishing, San Francisco, California, 1999.

[12] J. Xu and W. Croft. Corpus-based stemming using co-occurrence of word variants. *ACM Transactions on Information Systems*, 16(1):61–81, 1998.

[13] J. Xu, R. Weischdel, and C. Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of ACM SIGIR2001*, pages 105–109, 2001.

[14] C. Zhai and J. D. Lafferty. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of ACM SIGIR2001*, pages 111–119, 2001.

[15] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of ACM SIGIR2001*, pages 334–342, 2001.

[16] C. Zhai and J. D. Lafferty. Two-stage language models for information retrieval. In *Proceedings of ACM SIGIR2002*, pages 49–56, 2002.