Similarity Measures for Short Segments of Text

Donald Metzler[1], Susan Dumais[2], Christopher Meek[2]

[1]University of Massachusetts
Amherst, MA

[2]Microsoft Research
Redmond, WA

Abstract. Measuring the similarity between documents and queries has been extensively studied in information retrieval. However, there are a growing number of tasks that require computing the similarity between two very short segments of text. These tasks include query reformulation, sponsored search, and image retrieval. Standard text similarity measures perform poorly on such tasks because of data sparseness and the lack of context. In this work, we study this problem from an information retrieval perspective, focusing on text representations and similarity measures. We examine a range of similarity measures, including purely lexical measures, stemming, and language modeling-based measures. We formally evaluate and analyze the methods on a query-query similarity task using 363,822 queries from a web search log. Our analysis provides insights into the strengths and weaknesses of each method, including important tradeoffs between effectiveness and efficiency.

1 Introduction

Retrieving documents in response to a user query is the most common text retrieval task. For this reason, most of the text similarity measures that have been developed take as input a query and retrieve matching documents. However, a growing number of tasks, especially those related to web search technologies, rely on accurately computing the similarity between two very short segments of text. Example tasks include query reformulation (query-query similarity), sponsored search (query/ad keyword similarity), and image retrieval (query-image caption similarity).

Unfortunately, standard text similarity measures fail when directly applied to these tasks. Such measures rely heavily on terms occurring in both the query and the document. If the query and document do not have any terms in common, then they receive a very low similarity score, regardless of how topically related they actually are. This is well-known as the vocabulary mismatch problem. This problem is only exacerbated if we attempt to use these measures to compute the similarity of two short segments of text. For example, "UAE" and "United Arab Emirates" are semantically equivalent, yet share no terms in common.

Context is another problem when measuring the similarity between two short segments of text. While a document provides a reasonable amount of text to infer the contextual meaning of a term, a short segment of text only provides a limited context. For example, "Apple computer" and "apple pie" share the term apple, but are topically distinct. Despite this, standard text similarity measures would say that these

two short segments of text are very similar. However, computing the similarity between the query "Apple computer" and a full document about "apple pie" will produce a low similarity score since the document contains proportionally less text that is relevant to the query, especially compared to a full document about "Apple business news".

In this paper, we explore the problem of measuring similarity between short segments of text from an information retrieval perspective. Studies in the past have investigated the problem from a machine learning point of view and provided few, if any comparisons to standard text similarity measures. In this work, we describe a set of similarity measures that can be used to tackle the problem. These measures include simple lexical matching, stemming, and text representations that are enriched using web search results within a language modeling framework. In addition, we formally evaluate the measures for the query-query similarity task using a collection of 363,822 popular web queries. Our analysis provides a better understanding of the strengths and weaknesses of the various measures and shows an interesting tradeoff between effectiveness and efficiency.

The remainder of this paper is laid out as follows. First, Section 2 provides an overview of related work. We then describe the various ways to represent short segments of text in Section 3. Section 4 follows up this discussion by describing the similarity measures we investigated. Section 5 provides the details of our experimental evaluation on the query-query similarity task. Finally, in Section 6 we wrap up and provide conclusions and directions of future work.

2 Related Work

Many techniques have been proposed to overcome the vocabulary mismatch problem, including stemming [5,9], LSI [3], translation models [1], and query expansion [6,14]. This section describes several of these techniques that are most related to our work. The task we focus on is a query-query similarity task, in which we compare short text segments, such as "Apple computer", "apple pie", "MAC OS X", and "iMAC".

Translation models, in a monolingual setting, have been used for document retrieval [1], question answering [8], and detecting text reuse [7]. The goal is to measure the likelihood that some candidate document or sentence is a translation (or transformation) of the query. However, such models are less likely to be effective on very short segments of texts, such as queries, due to the difficulty involved in estimating reliable translation probabilities for such pieces of text.

Query expansion is a common technique used to convert an initial, typically short, query into a richer representation of the information need [6,10,14]. This is accomplished by adding terms that are likely to appear in relevant or pseudo-relevant documents to the original query representation. In our query-query matching work, we explore expanding both the original and candidate query representations.

Sahami and Heilman proposed a method of enriching short text representations that can be construed as a form of query expansion [11]. Their proposed method expands short segments of text using web search results. The similarity between two short segments of text can then computed in the expanded representation space. The expanded representation and DenseProb similarity measure that we present in

Sections 3 and 4 are similar to this approach. However, we estimate term weights differently and analyze how such expansion approaches compare, in terms of efficiency and effectiveness, to other standard information retrieval measures.

Finally, since we evaluate our techniques on a query-query similarity task, it should be noted that this problem, and the related problem of suggesting and identifying query-query reformulations has been investigated from a number of angles, ranging from machine learning approaches [4] to query session log analysis[2]. These techniques are complimentary to the core representational and similarity ideas that we explore in our work.

3 Text Representations

Text representations are an important part of any similarity measure. In this section, we describe three different ways of representing text. Although these representations can be applied to text of any length, we are primarily interested in using them to represent short segments of text.

3.1 Surface Representation

The most basic representation of a short segment of text is the surface representation (i.e. the text itself). Such a representation is very sparse. However, it is very high quality because no automatic or manual transformations (such as stemming) have been done to alter it. While it is possible that such transformations enhance the representation, it is also possible that they introduce noise.

3.2 Stemmed Representation

Stemming is one of the most obvious ways to generalize (normalize) text. For this reason, stemming is commonly used in information retrieval systems as a rudimentary device to overcome the vocabulary mismatch problem. Various stemmers exist, including rule-based stemmers [9] and statistical stemmers [5].

Although stemming can significantly improve matching coverage, it also introduces noise, which can lead to poor matches. Using the Porter stemmer, both “marine vegetation” and “marinated vegetables” stem to “marin veget”, which is undesirable. Overall, however, the number of meaningful matches introduced typically outweighs the number of spurious matches.

Throughout the remainder of this paper, we use the Porter stemmer to generate all of our stemmed representations.

3.3 Expanded Representation

Although stemming helps overcome the vocabulary mismatch problem to a certain extent, it does not handle the contextual problem. It fails to discern the difference between the meaning of “bank” in “Bank of America” and “river bank”. Therefore, it

<query>*apple pie*</query>

<title>Applie pie – Wikipedia, the free encyclopedia</title>
<snippet>In cooking, an apple pie is a fruit pie (or tart) in which the principal filling ingredient is apples . Pastry is generally used top-and-bottom, making a double-crust pie, the upper crust of which ...</snippet>
<url>en.wikipedia.org/wiki/Apple_ pie</url>

<title>All About Food – Apple Pies</title>
<snippet>Apple Pie. Recipes. All-American Apple Pie. American Apple Pie. Amish Apple Pie . Apple Cream Pie. Apple Crumble Pie. Apple Pie . Apple Pie in a Brown Bag. Best Apple Pie</snippet>
<url>fp.enter.net/~rburk/pies/ applepie/applepie.htm</url>

<title>Apple Pie Recipe</title>
<snippet>Apple Pie Recipe using apple peeler corer slicer ... Apple Pie Recipe. From Scratch to Oven in 20-Minutes. Start by preheating the oven. By the time it's ...</snippet>
<url>applesource.com/applepierecipe.htm</url>
…

Fig. 1. Example expanded representation for the text "apple pie." The expanded representation is the concatenation of the title and snippet elements.

is desirable to build representations for the short text segments that include contextually relevant information.

One approach is to enrich the representation using an external source of information related to the query terms. Possible sources of such information include web (or other) search results returned by issuing the short text segment as a query, relevant Wikipedia articles, and, if the short text segment is a query, query reformulation logs. Each of these sources provides a set of contextual text that can be used to expand the original sparse text representation.

In our experiments, we use web search results to expand our short text representations. For each short segment of text, we run the query against a commercial search engine's index and retrieve the top 200 results. The titles and snippets associated with these results are then concatenated and used as our expanded representation. In Figure 1, we show a portion of the expanded representation for the short text segment "apple pie". As we see, this expanded representation contains a number of contextually relevant terms, such as "recipe", "food", and "cooking" that are not present in the surface representation. We note that this expanded representation is similar to the one proposed in [11].

4 Similarity Measures

In this section we describe three methods for measuring the similarity between short segments of text. These measures are motivated by, and make use of, the representations described in the previous section. We also propose a hybrid method of combining the ranking of the various similarity measures in order to exploit the strengths and weaknesses of each.

4.1 Lexical

The most basic similarity measures are purely lexical. That is, they rely solely on matching the terms present in the surface representations. Given two short segments of text, Q and C, treating Q as the query and C as the candidate we wish to measure the similarity of, we define the following lexical matching criteria:

- Exact – Q and C are lexically equivalent. (Q: "seattle mariners tickets", C: "seattle mariners tickets")
- Phrase – C is a substring of Q. (Q: "seattle mariners tickets", C: "seattle mariners")
- Subset – The terms in C are a subset of the terms in Q. (Q: "seattle mariners tickets", C: "tickets seattle")

These measures are binary. That is, two segments of text either match (are deemed 'similar') or they do not. There is no graded score associated with the match. However, if necessary, it is possible to impose such a score by looking at various characteristics of the match such as the length of Q and C, or the frequency of the terms in some collection.

It should also be noted that exact matches $\subseteq$ phrase matches $\subseteq$ subset matches. Exact matches are very high precision (excellent matches), yet very low recall since they miss a lot of relevant material. At the other extreme, subset matches are lower precision, but have higher recall. Any candidate C that contains a term that does not appear in the query Q will not match under any of these rules, which is very undesirable. Therefore, we expect that matches generated using these lexical rules will be have high precision but poor recall.

4.2 Probabilistic

As we just described, lexical matching alone is not enough to produce a large number of relevant matches. In order to improve recall, we must make use of the expanded text representations. To do so, we use the language modeling framework to model query and candidate texts.

To utilize the framework, we must estimate unigram language models for the query (θ_Q) and each candidate (θ_C). For ranking purposes, we use the negative KL-divergence between the query and candidate model, which is commonly used in the language modeling framework [14]. This results in the following ranking function:

$$\begin{aligned} -KL(\theta_Q, \theta_C) &= H(\theta_Q) - CE(\theta_Q, \theta_C) \\ &\equiv \sum_{w \in V} P(w \mid \theta_Q) \log P(w \mid \theta_C) \end{aligned} \quad \textbf{(1)}$$

where V is the vocabulary, H is entropy, CE is cross entropy, and $\equiv$ denotes rank equivalence.

The critical part of the ranking function is how the query and candidate language models are estimated. Different estimates can lead to radically different rankings. We now describe how we estimate these models using the representations available to us.

We begin with the query model. The most straightforward way of estimating a query model is to use the surface representation. This is estimated as:

$$P(w \mid \theta_Q) = \frac{tf_{w,QS}}{|QS|} \quad (2)$$

where QS denotes the query surface representation, $tf_{w,QS}$ is the number of times w occurs in the representation, and $|QS|$ is the total number of terms in QS. This estimate will be very sparse since we are using the surface representation. This allows Equation 1 to be computed very efficiently since most terms in the summation ($w \in V$) will be zero.

We also consider the case when we use the expanded representation of the query, as described in Section 3.3. The estimate, which is analogous to the unexpanded case, is:

$$P(w \mid \theta_Q) = \frac{tf_{w,QE} + \mu_Q P(w \mid C)}{|QE| + \mu_Q} \quad (3)$$

where QE is the query expanded representation, and μ_Q is a smoothing parameter. This type of estimation is commonly used in the language modeling community and is often referred to as Dirichlet or Bayesian smoothing [13]. Since this estimate is much more dense than the unexpanded estimate, it is more time consuming to evaluate Equation 1. Due to the amount of data we work with in our experiments, we truncate this distribution by only keeping the 20 most likely terms and setting the remaining probabilities to 0. Pruning similar to this was done in [11] for the same reason.

Finally, we describe how the candidate model is estimated. Rather than exploring both estimates using both unexpanded and expanded representations, we restrict ourselves to expanded representations. Therefore, we get the following estimate:

$$P(w \mid \theta_C) = \frac{tf_{w,CE} + \mu_C P(w \mid C)}{|CE| + \mu_C} \quad (4)$$

where CE is the candidate expanded representation, and μ_C is a smoothing parameter. Unlike the expanded query model, we do not truncate this distribution in any way.

Finally, it is important to recall that expanded representations may be created using any number of external sources. Our use of the web was simply a matter of convenience. However, we can use this same general framework with expanded representations generated using any possible external text source.

4.3 Hybrid

We are often interested in taking the matches generated by several different similarity measures and combining them. We call these *hybrid* techniques. Given two or more lists of matches, we stack the lists according to some pre-defined ordering (denoted ">") of the lists, to form a combined list. For example, given match lists A and B, and ordering A > B, we form the hybrid list AB, which is list B appended to the end

Method Name	*Query Representation*	*Candidate Representation*	*Similarity Measure*
Lexical	Surface	Surface	Hybrid (Exact > Phrase > Subset)
Stemming	Stemmed	Stemmed	Hybrid (Lexical > Exact Stems)
SparseProb	Surface	Expanded	Probabilistic
DenseProb	Expanded	Expanded	Probabilistic
Backoff	Various	Various	Hybrid (Exact > Exact Stems > DenseProb)

Table 1. Overview of query representation, candidate representation, and similarity measure used for each matching method.

of list A. Since the same match may occur in more than one set of results, we must remove duplicates from the combined list. Our deduplication policy states that we keep the highest ranked match and remove all others. Although this combination scheme is naïve, it has the advantage that there are no combination parameters to learn.

4.4 Summary of Methods Evaluated

Table 1 summarizes the methods we evaluate in the next section. For each method, we include the query and candidate representations and the similarity measure used.

The Lexical method, which considers the surface forms of the query and candidate, makes use of a hybrid technique that ranks exact matches first, then phrase matches, and finally subset matches. The Stemming method also uses a hybrid technique that first ranks matches using the Lexical method just described and then ranks any exact matches that result after stemming both the query and the candidate. We refer to these types of matches as "exact stems" matches.

The SparseProb method is the first of the two probabilistic methods. It uses the unexpanded query representation, the expanded candidate representation, and ranks using the negative KL-divergence, whereas the DenseProb method uses expanded representations for both the query and the candidate and also ranks using the negative KL-divergence.

Finally, the Backoff method is a hybrid method that ranks exact matches, exact stems matches, and then DenseProb matches. The goal here is to see what benefit, if any, is achieved by replacing the phrase and subset matches from the Stemming method with DenseProb matches. We hypothesize that the DenseProb matches will be better than the often poor phrase and subset matches.

Many other query/candidate representation combinations are possible beyond those listed in Table 1. For example, it may be reasonable to use an expanded query form and a surface candidate form. However, in order to maintain a reasonable scope, we constrain ourselves to the methods described in this section.

Query: "seattle mariners"				
Lexical	*Stemming*	*SparseProb*	*DenseProb*	*Backoff*
seattle mariners	seattle mariners	seattle mariners tickets	seattle mariners tickets	seattle mariners
seattle	seattle	mariners tickets	mariners tickets	seattle mariner
mariners	mariners	seattle mariners	seattle mariners baseball	seattle mariners tickets
	seattle mariner	seattle mariners baseball	seattle mariners	mariners tickets
		seattle mariners schedule	seattle mariners schedule	seattle mariners baseball
		mariners baseball	mariners baseball	seattle mariners schedule
		seattle baseball	seattle baseball	mariners baseball
		mariners	red sox mariners tickets	seattle baseball
		mariners schedule	mariners schedule	red sox mariners tickets
		seattle mariner	cheap mariners tickets	mariners schedule

Table 2. Examples matches taken from our test collection for the query "seattle mariners". The Seattle Mariners are a baseball team from Seattle. For each method, we show the 10 matches with the highest similarity score.

5 Experimental Evaluation

In this section we evaluate the similarity measures proposed in Section 4. We begin by showing some illustrative examples of matches generated using our algorithms. We then formally evaluate the methods in the context of a query-query similarity task.

5.1 Illustrative Examples

Table 2 provides illustrative matches returned using the various matching techniques described in Section 4. Although many of these results look reasonable, it is difficult to quantify how much better any one method is by simply looking at these results. Therefore, in the next section we formally evaluate the different match types.

5.2 Query-Query Similarity

We now describe our query-query similarity experiments. Here, we are interested in evaluating how well the various methods we described in Section 4 can be used to find queries that are similar to some target query. This task is a general task that is widely applicable. For example, such a query-query similarity system could be used to recommend alternative queries to users of a web search engine or for session boundary detection in query log analysis.

Judgment	*Description*	*Examples (Query / Candidate)*
Excellent	The candidate is *semantically equivalent* to the user query.	atlanta ga / atlanta georgia
Good	The candidate is related to (but not identical to) the query intent and it is likely the *user would be interested in the candidate*.	seattle mariners / seattle baseball tickets
Fair	The candidate is related to the query intent, but in an overly vague or specific manner that results in the *user having little, if any, interest in the candidate*.	hyundia azera / new york car show
Bad	The candidate is *unrelated* to the query intent.	web visitor count / coin counter

Table 3. Description of the relevance judgment scale.

5.2.1 Data

The following data resources were used in our experimental evaluation. A sample of 363,822 popular queries drawn from a 2005 MSN Search query log was used as our candidate pool of queries to match against. For each query, we generated an expanded representation, as described in Section 3.3. In our experiments, we set μ_Q to 0 and μ_C to 2500. To handle this amount of data, we built an index out of the expanded representations using the Indri search system [12].

We also randomly sampled a set of 120 queries from the same log to use as target queries. These target queries were then matched against the full set of 363k queries. For each of these target queries, we ran the methods described in Section 4 and pooled the results down to a depth of 25 per method. A single human assessor then judged the relevance of each candidate result with respect to the target query using a 4-point judgment scale. Table 3 provides a description and examples of each type of judgment.

The result of this assessment was 5231 judged target/candidate pairs. Of these judgments, 317 (6%) were Excellent, 600 (11%) were Good, 2537 (49%) were Fair, and 1777 (34%) were Bad. In order to determine the reliability of the judgments, four assessors judged 10 target queries. The inter-annotator agreement was then computed for these queries and was found to be 60%. However, when Excellent and Good judgments were binned and Fair and Bad judgments were binned, the agreement increased to 80%. This indicates the boundary between Fair and Bad is interpreted differently among users. For this reason, we will primarily focus our attention on the boundary between Excellent and Good and between Good and Fair. In addition, the Excellent and Good matches are the most interesting for many practical applications including query suggestion and sponsored search.

5.2.2 Evaluation

We are interested in understanding how our matching methods compare to each other across various relevance criteria. Since we are interested in using standard information retrieval metrics, such as precision and recall, we must binarize the relevance judgments. For each experiment, we state the relevance criteria used.

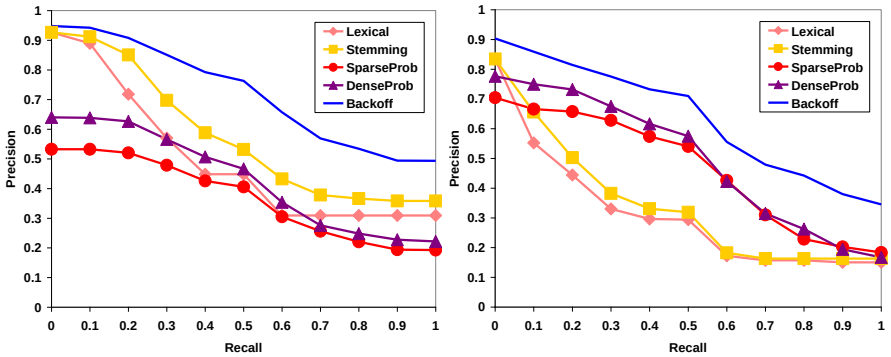

Fig. 2. Interpolated, 11-point precision-recall curves for the five matching methods described in Section 4. On the left, candidates judged 'Excellent' are considered relevant. On the right, candidates judged 'Excellent' or 'Good' are considered relevant.

We first evaluate the methods using precision-recall graphs using two different relevance criteria. The results are given in Figure 2. For the case when Excellent matches are considered relevant (left panel), we see that the Lexical and Stemming methods outperform the probabilistic methods, especially at lower recall levels. This is not surprising, since we expect lexical matches to easily find most of the Excellent matches. In addition, we see that Stemming consistently outperforms the Lexical method. However, the Backoff method dominates the other methods at all recall levels. This results from backing off from stricter matches to less strict matches. For example, for the query "atlanta ga", the Lexical method will match "atlanta ga", but neither the Lexical nor the Stemming methods will match "atlanta georgia", which is actually an Excellent match that is found using the DenseProb method.

When we relax the relevance criteria and consider both Excellent and Good judgments to be relevant (right panel), we see an interesting shift in the graph. Here, the probabilistic methods, SparseProb and DenseProb, outperform the Lexical and Stemming methods at all recall levels, except very low levels. This suggests that the Lexical and Stemming methods are good at finding Excellent matches, but that they are worse at finding Good matches compared to the probabilistic methods. We further test this hypothesis later in this section. However, once again, we see that the Backoff method outperforms all of the methods at all recall levels.

One reason why the Backoff method is superior to the non-hybrid probabilistic methods is the fact that the SparseProb and DenseProb methods often fail to return exact matches high in the ranked list. This is caused by truncating the expanded query distribution before computing the KL divergence. Since exact matches account for a majority of the Excellent judgments, this causes the entire curve to be shifted down. By forcing the exact and exact stems matches to occur first, we are 'stacking the deck' and promoting matches that are likely to be high precision. This, combined with the high recall of the DenseProb method, results in a superior matching method.

It is clear that exact matches are very likely to result in Excellent matches. However, it is not clear how phrase and subset lexical matches compare to stemming and probabilistic matches. To measure this, we compute the precision at k for the Lexical and Backoff methods, where k is the number of results returned by the

		R = {Excellent}		R = {Excellent, Good}	
k	Queries	Lexical	Backoff	Lexical	Backoff
1	40	0.7500	0.8125	0.7500	0.8125
2	38	0.3235	0.4853	0.3382	0.5882
3	31	0.2688	0.4194	0.3978	0.5914

Table 3. Precision at *k*, where *k* is the number of matches returned using the Lexical method. In this table, the evaluation set of queries was stratified according to *k*. Queries indicates the the number of queries associated with each *k* . Only values of *k* associated with 10 or more queries are shown.

Lexical method. This evaluation allows us to quantify the improvement achieved by replacing the low precision phrase and subset matches with the high precision exact stems matches and high recall DenseProb matches.

The results are presented in Table 4 for two relevance criteria. We stratify the queries with respect to *k*, the number of Lexical method matches for the query, and compute precision at depth *k* over these queries. We only include values of *k* associated with 10 or more queries, since it misleading to compute and compare means over smaller samples. As the results show, the Backoff method is superior in every case. This suggests that the stemming and probabilistic matches (used in the Backoff method) are considerably better at finding both Excellent and Good matches compared to the phrase and subset matches (used in the Lexical method).

5.2.3 Effectiveness vs. Efficiency

One important practical aspect of the techniques developed is efficiency. Generating lexical and stemming matches is very efficient. The probabilistic methods are slower, but not unreasonable. Generating matches against our collection of 363,822 candidates using a modern single CPU machine takes 0.15 seconds per query using the SparseProb method and 3 seconds per query using the DenseProb method.

The DenseProb method requires, *a priori*, an index of expanded representations for both the candidates and the incoming queries. If we are asked to generate DenseProb matches for a query that is not in our index, then we must generate this representation on the fly. However, the SparseProb method does not exhibit this behavior and can be used to efficiently generate matches for *any* incoming query.

Therefore, SparseProb is the the best choice in terms of speed and coverage. However, if speed is not an issue, and high quality results are important, then DenseProb is the better choice.

6 Conclusions and Future Work

In this paper we studied the problem of measuring the similarity between short segments of text. We looked at various types of text representations, including surface, stemmed, and expanded. We showed how web search results can be used to form expanded representations of short text segments. We then described several

similarity measures based on these representations, including lexical matching and probabilistic measures based on language models estimated from unexpanded and expanded representations. We then formally evaluated and compared these measures in the context of a query-query similarity task over a large collection of popular web search queries. Our results showed that lexical matching is good for finding semantically identical matches and that the probabilistic methods are better at finding interesting topically related matches. It was shown that a simple hybrid technique that combines lexical, stemmed, and probabilistic matches results in far superior performance than any method alone.

The probabilistic framework presented in this paper provides a general method for measuring the similarity between two short segments of text. Although we chose to use web search results as the basis of our expanded representation in this work, an interesting direction of future work would be to use a variety of other sources of external text, such as query reformulation logs, queries that result in similar click patterns, and Wikipedia. It would also be worthwhile to evaluate these techniques in an end-to-end application, such as a query-query reformulation system, in order to see what impact they have in a more practical setting.

References

[1] Berger, A. and Lafferty, J. Information retrieval as statistical translation. In *Proceedings of SIGIR '99*, pages 222-229, 1999.

[2] Cucerzan, S. and Brill, E. Extracting semantically related queries by exploiting user session information. Technical Report, Microsoft Research, 2005.

[3] Deerwester, S., Dumais, S., Landauer, T., Furnas, G. and Harshman, R. Indexing by latent semantic analysis. In *JASIST*, 41(6), pages 391-407, 1990.

[4] Jones, R., Rey, B., Madani, O., and Greiner W. Generating query substitutions. In *Proceedings of WWW 2006*, pages 387-396, 2006.

[5] Krovetz, R. Viewing morphology as an inference process. In *Proceedings of SIGIR '93*, pages 191-202, 1993.

[6] Lavrenko, V. and Croft, W.B. Relevance based language models. In *Proceedings of SIGIR '01*, pages 120-127, 2001.

[7] Metzler, D., Bernstein, Y., Croft, W.B., Moffat, A., and Zobel, J. Similarity measures for tracking information flow. In *Proceedings of CIKM '05*, pages 517-524, 2005.

[8] Murdock, V. and Croft, W.B. A Translation Model for Sentence Retrieval. In *Proceedings of HLT/EMNLP '05*, pages 684-691, 2005.

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

[10] Rocchio, J. J. *Relevance Feedback in Information Retrieval*, pages 313-323. Prentice-Hall, 1971.

[11] Sahami, M. and Heilman, T. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of WWW 2006*, pages 377-386, 2006.

[12] Strohman, T., Metzler, D., Turtle, H., Croft, W. B. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligence Analysis*, 2005.

[13] Zhai, C. and Lafferty, J. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR '01*, pages 334-342, 2001.

[14] Zhai, C. and Lafferty, J. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM '01*, pages 403-410, 2001.