Improving Similarity Measures for Short Segments of Text

Wen-tau Yih and **Christopher Meek**
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{scottyih, meek}@microsoft.com

Abstract

In this paper we improve previous work on measuring the similarity of short segments of text in two ways. First, we introduce a Web-relevance similarity measure and demonstrate its effectiveness. This measure extends the Web-kernel similarity function introduced by Sahami and Heilman (2006) by using relevance weighted inner-product of term occurrences rather than TF×IDF. Second, we show that one can further improve the accuracy of similarity measures by using a machine learning approach. Our methods outperform other state-of-the-art methods in a general query suggestion task for multiple evaluation metrics.

Introduction

The problem of measuring the similarity between two very short text segments has become increasingly important for many Web-related tasks. Examples of such tasks include *query reformulation* (similarity between two queries), *search advertising* (similarity between the user's query and advertiser's keywords), and product keyword recommendation (similarity between the given product name and suggested keyword).

Measuring the semantic similarity between two texts has been studied extensively in the IR and NLP communities. However, the problem of assessing the similarity between two *short* text segments poses new challenges. Text segments commonly found in these tasks range from a single word to a dozen words. Because of the short length, the text segments do not provide enough context for surface-matching methods such as computing the cosine score of the two text segments to be effective. On the other hand, because many text segments in these tasks contain more than one or two words, traditional corpus-based *word* similarity measures can fail too. These methods typically rely on the co-occurrences of the two compared text segments and, because of their lengths, they may not co-occur in any documents even when using the whole Web as the corpus. Finally, because of the diversity of the text segments used in these Web applications, linguistic thesauruses such as WordNet do not cover a significant fraction of the input text segments. In order to overcome these difficulties, researchers have recently proposed several new methods for measuring similarity of short text segments (Sahami & Heilman 2006; Jones *et al.* 2006; Metzler, Dumais, & Meek 2007).

In this paper, we study the problem of measuring similarity of short text segments in a general query suggestion scenario: given a short text segment q and a list of suggestions $\{s_1, s_2, ..., s_n\}$, we would like to rank suggestions based on their similarity to q or select a subset of suggestions that are similar to q. Our contributions are as follows. First, we introduce a web-relevance similarity measure which improves the web-based kernel method (Sahami & Heilman 2006) through a new term weighting scheme. Instead of using the traditional TF×IDF score or its variations, we use the "relevancy" of the words to the document, estimated by a state-of-the-art keyword extractor (Yih, Goodman, & Carvalho 2006). Second, in order to leverage the strengths of different similarity measures, we propose to combine them using machine learning. In particular, we consider two learning approaches: one directly models the similarity between a query and a suggestion (q, s_i) and the other models the preference ordering between two suggestions s_i and s_j, with respect to the same query q. Finally, we present an experimental comparison between existing approaches for measuring similarity between short text segments and our enhanced similarity measures. The experiments indicate that our methods are significantly better than existing methods.

The rest of the paper is organized as follows. We first review existing methods for measuring similarity of short text segments. We then introduce our Web-relevance similarity measure and the proposed learning approaches, followed by the experimental evaluation.

Related Work

We review existing methods for measuring similarity between short text segments that are very related to our approaches. We consider three categories of methods: *surface matching*, *corpus-based methods* and *query-log methods*.

Surface Matching

Given an input query and suggestion (q, s), the idea of surface-matching methods is based on the number of words that occur in both text segments. Suppose Q and S are the sets of words in q and s, respectively. Common similarity measures discussed in (Manning & Schütze 1999) are listed as follows.

Matching	$\|Q \cap S\|$
Dice	$2\|Q \cap S\|/(\|Q\| + \|S\|)$
Jaccard	$\|Q \cap S\|/\|Q \cup S\|$
Overlap	$\|Q \cap S\|/\min(\|Q\|, \|S\|)$
Cosine	$\|Q \cap S\|/\sqrt{\|Q\| \times \|S\|}$

There are several variations of these surface-matching methods. For example, the sets Q and S can be constructed using the stemmed words instead of the original words. Two words can be matched if they are synonyms according to some thesauruses such as WordNet. The set operations described here are equivalent to vector operations when representing q and s as binary vectors, where an element indicates whether a word in the vocabulary appears in the original text segments. It is therefore straightforward to extend these measures as operations of real-valued vectors, where each element may represent the frequency of the word.

Although different statistics for surface matching have their own strengths and weaknesses, their quality on measuring the similarity of very short text segments is usually unreliable (Sahami & Heilman 2006). Therefore, different expanded representations of the original text segments have been proposed to replace the surface vectors.

Corpus-based Methods

One method to overcome the weakness of surface matching is to leverage the information derived from a large corpus, which is often the Web. There are two kinds of popular corpus-based methods. One focuses on using the information of whether the text q and s co-occur in the same document. Representative methods used for measuring semantic similarity of words include pointwise mutual information (Turney 2001), latent semantic analysis (Landauer, Foltz, & Laham 1998) and normalized document set overlap (Fitzpatrick & Dent 1997). While these methods can potentially be applied to general short text segments, as the lengths of the text segments increase, the chance that these two text segments co-occur in some documents decreases substantially, which affects the quality of these similarity measures.

The other kind of corpus-based methods use the large document collection to represent each of the text segments separately. A typical approach is to use the words in the documents that contain the text segments as the expanded representation, and the similarity score is then calculated using the new representations. For example, measuring the similarity of the word distributions between the expanded representations using KL-divergence has been proposed by Metzler, Dumais, & Meek (2007) recently. Another type of methods treat the expanded representation as a sparse vector and adapt operations used for surface matching. For instance, Sahami & Heilman (2006) created a web-based kernel function, which basically constructs two unit vectors using the words in the expanded representations and then returns the inner-product as the final similarity score. Because we will show how to improve this Web kernel similarity measure, we describe more details of this method here.

Web-based Similarity Kernel (Sahami & Heilman 2006) Given a pair of short text segments q and s, the Web-based similarity kernel is simply the inner-product of the expanded vector representations, denoted as $QE(q) \cdot QE(s)$. The query expansion representation QE of a short text segment x is derived as follows.

1. Let $D_n(x)$ be the set of top n documents returned by a search engine when using x as the query term.
2. For each document $d_i \in D_n(x)$, construct the term vector v_i, where each element is the weighted score $w_{i,j}$ of term t_j, defined as follows.
$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_j}), \quad (1)$$
where $tf_{i,j}$ is the term frequency of t_j in d_i, N is the total number of documents in the corpus for calculating document frequencies, and df_j is the document frequency of term t_j.
3. Let $C(x)$ be the centroid of the L_2 normalized vectors v_i: $C(x) = \frac{1}{n}\sum_{i=1}^{n} v_i/||v_i||$
4. Let $QE(x)$ be the L_2 normalization of the centroid $C(x)$: $QE(x) = C(x)/||C(x)||$

This method can be modified to improve efficiency. One example is to consider only high weighted terms in the vectors. Another example is to use only the summary generated by the search engine to represent the whole document d_i. Both techniques were implemented in their system (Sahami & Heilman 2006).

Query-log Methods

Commercial search engines such as Google or Yahoo receive millions of queries per day. The search query logs have become a great resource for measuring similarity between short text segments, especially for tasks such as query suggestion. One recent example is the work of generating query substitutions by Jones *et al.* (2006). In this task, their goal is to generate alternative query suggestions to a given query. They first generated suggestions based on the information about whether the target query and suggestion had appeared in the same session query log. These suggestions were then ranked based on a linear regression model trained with three major types of features: *surface characteristics* such as number of characters or words of the query and suggestion, *syntactic difference* between the query and suggestion such as Levenshtein edit distance or size of prefix overlapping, and *substitution statistics* such as the log-likelihood ratio or the mutual information between the query and suggestion using their distributions in the query logs. Probably due to the fact that candidate suggestions were selected using the substitution statistics, they found that the only useful features here were based on syntactic differences. In addition, sophisticated learning methods such as linear SVMs and decision trees did not outperform the simple linear regression model.

Compared to previous methods described in this section, Jones *et al.* did not aim to provide a similar metric. Measuring the similarity between the query and suggestion is somewhat bound to the generation task. In addition, the coverage for pairs of short text segments is limited because subsets of the words in both segments must appear in the same user session query logs.

Web-relevance Similarity Measure

In this section, we describe a Web-relevance similarity measure, which extends the Web-based similarity kernel function by using a better term weighting scheme.

The web-based similarity kernel function uses term and document frequencies to measure the importance of the terms in the expanded representation of the input text segment (Eq. 1). While using TF×IDF or its variations is a simple and effective method to evaluate the importance of a word in a given document, this crude measure may not always be reliable. For example, a word that appears in the beginning of a document is typically more important and relevant to the topic of the document, and words that appear in the document title should not be treated the same as words in the body. However, this kind of information is not captured by either term frequency or document frequency. Another potential problem is the side effect of using document frequency. Document frequency is very effective in down-weighting stopwords, which have both high TF and DF values. However, not all the high-DF words are useless. Popular keywords that interest people may be broadly discussed in many different documents, which can be easily down-weighted by their high-DF values, even though they are actually important[1].

Because of these weaknesses of using only TF and DF, we use an alternative approach to assessing the importance of terms in a document. In particular, we consider a keyword extraction approach to term weighting. In this approach, a keyword extraction system is used to associate a relevance score with terms and phrases extracted from a document (Frank *et al.* 1999; Turney 2000; 2003; Goodman & Carvalho 2005; Yih, Goodman, & Carvalho 2006). We use the relevance score output by a keyword extraction system as the weighting function in our Web-relevance similarity measure.

We first build a keyword extraction system following the approach described in (Yih, Goodman, & Carvalho 2006), which is the current state-of-the-art keyword extraction system trained using more than 10 categories of features. In their original design, all the phrases (i.e., consecutive words) up to length 5 are treated as keyword candidates. Because the goal here is to judge the importance of the words in the document, we consider only words as candidates. For efficiency reason, we also use the summary pages generated by the search engine as the document, which is similar to the technique used in (Sahami & Heilman 2006; Metzler, Dumais, & Meek 2007). The detailed steps of the new query expansion method are described below.

[1]In the 222,867 pages we sampled from the web for deriving document frequency, we observe several terms that have high-DF values but are also popular search queries. Examples of this sort include *travel*, *books*, *jobs*, *casino*, and names of big corporations such as *UPS*, *AOL*, *Monster* and *CNN*.

1. Let $D_n(x)$ be the set of top n documents returned by a search engine when using x as the query term.
2. Construct a document d by concatenating the title and short summary of each document $d_i \in D_n(x)$.
3. Construct the term vector v, where each element is the relevancy score w_j output by the keyword extraction system.
4. Let $QE_{rel}(x)$ be the L_2 normalization of the vector v: $QE_{rel}(x) = v/||v||$.

Similarly, given a pair of short text segments q and s, the Web-relevance similarity measure is $QE_{rel}(q) \cdot QE_{rel}(s)$.

Learning Similarity Measures

In this section, we develop a machine learning approach that uses labeled training data to improve the similarity measure for short text segments. Existing similarity measures for short text segments, including our Web-relevance similarity measure, typically suffer from two limitations. First, these measures are *static* measures of similarity given a corpus. There is no reason to believe that any single static measure is ideal for all applications. To the contrary, the existence of a variety of alternative similarity function is evidence that the ideal similarity measure is a function of the target application. Second, different similarity measures have different *coverage*. For example, query-log methods cover only the queries that appear in the search query logs; Web-based kernel or Web-relevance function have difficulties providing robust similarity measures for new or rarely used text segments, those which occur in very few pages indexed by a search engine.

Recently, researchers have started to address these two issues at least implicitly. For example, a learning approach has been used by Jones *et al.* (2006) to fine tune their system to suggest better query alternatives. The coverage issue has been improved using a hybrid approach by Metzler, Dumais, & Meek (2007) in which a simple rule is used to combine some surface-matching methods with their KL-divergence similarity measure. However, none of these approaches covers both limitations in a principled way.

In this section, we propose using machine learning to improve the similarity measure. By taking the output of existing similarity measures as features, we can effectively combine them to increase the overall coverage. Furthermore, by using application-specific labeled data to train the model, the similarity measure can be tuned to the target application.

Learning Approaches

While there are other alternative approaches such as ordinal regression (Herbrich, Graepel, & Obermayer 2000), in this paper, we consider two machine learning approaches for learning better similarity measure for short text segments.

In the first approach we learn the *similarity metric* directly. Given a pair of text segments q and s, the goal is to learn a monotonic function $f_m : (q,s) \rightarrow \mathbf{R}$. $f_m(q_i, s_i) > f_m(q_j, s_j)$ indicates that q_i and s_i are more alike compared to q_j and s_j. Training this model requires labels for a set of text segment pairs. When specific numeric similarities scores are given, a regression function can be learned. When only binary labels that indicate whether a pair of text segments is similar or not are given, a probability estimator can be learned, and the probability that the given text segments q and s are similar (i.e., $P(sim(q,s)=1)$) can be used as the similarity metric directly.

In the second approach we learn *preference ordering*. In many applications, including those considered in this paper, the goal is to obtain a *ranked* set of candidates. We consider learning to predict the *pairwise preference* of each pair of suggested candidates s_i and s_j, with respect to the same query q. It is important to note that a set of preference orderings need not be consistent with a total ordering. We use a simple weighted voting mechanism to score each candidate and obtain a total ordering. Learning preference ordering has been advocated by researchers (e.g., Burges *et al.* 2005) and is motivated by the observation that preference annotations are generally more reliable than categorical similarity labels. When given the labeled preference for each pair of suggestions, we can train a probabilistic binary classifier to predict whether s_i is a more preferable suggestion than s_j.

One can apply a variety of alternative learning algorithms for these two approaches. Since comparing different learning algorithms is not our goal, in this paper, we choose logistic regression for its good empirical performance and clear probability interpretation of its output [2]. In principle, other learning algorithms can be used as well.

Because one of the main reasons for using a machine learning approach is to improve the coverage of static similarity measures, we use the output of different similarity measures and some variations such as the derived ranks as features. These features include all the surface-matching methods described previously (i.e., *matching*, *Dice coefficient*, *Jaccard coefficient*, *overlap* and *cosine*), the KL-divergence method (Metzler, Dumais, & Meek 2007), Web-based similarity kernel (Sahami & Heilman 2006) and our Web-relevance similarity measure. Clearly, additional features can be considered including other properties of the short text segments, such as their query log frequencies, length of the segments, word edit distance between and more. To simplify the presentation, however, we only describe results of using static similarity features.

Experiments

In this section, we describe the experimental comparison between our methods and alternative state-of-the-art methods. We describe the data that we collected for a general query suggestion task and the results of applying various similarity measures to this data. We demonstrate that our Web-relevance function outperforms existing corpus-based methods and better ranking results can be achieved by combining various similarity measures using our learning approaches.

[2]The actual training method used in experiments is the SCGIS algorithm (Goodman 2002). The variance of the Gaussian prior is 3 and the number of iterations is 100.

	P_a	P_e	κ
Excellent, Good, Fair, Bad	0.598	0.397	0.338
(Excellent & Good) vs. (Fair & Bad)	0.844	0.723	0.435
Preference ($\rhd$, $\equiv$, $\lhd$)	0.647	0.414	0.399
Preference ($\unrhd$, $\lhd$)	0.827	0.673	0.473
Preference ($\unlhd$, $\rhd$)	0.799	0.645	0.432

Table 1: The inter-annotator agreement analysis on different groupings of classification and preference choices

Data

We created a query suggestion data set in the following way. We first built our query and suggestion candidate set by taking a random sample of 363 thousand queries from the top 1 million most frequent queries in late 2005. Among those candidates, 122 queries were randomly selected as our target queries. For each target query, up to 100 queries that were considered relevant (according to a set of alternative mechanisms) were used as the suggestions.

Human annotators then judged the degree of similarity of each query and suggestion, and labeled it using a 4-point scale – *Excellent*, *Good*, *Fair* and *Bad*. The annotator guidelines indicated that Excellent and Good should be used when the suggested keywords are clearly related to the query intent, while Fair and Bad should be used when the suggested keywords are too general or unrelated. We managed to collect 4,852 labeled query/suggestion pairs. Some of the pairs are labeled by more than one annotator for the analysis of inter-annotator agreement. When these query/suggestion pairs are used for evaluating different similarity measures, the final label is the class that has the most votes. Of these effective annotations, the ratios of the four labels are: Excellent - 5%, Good - 12%, Fair - 44% and Bad - 39%.

To better understand the degree to which similarity judgements are subjective we evaluate the inter-annotator agreement for our data set. One standard approach for assessing the inter-annotator agreement is the the Kappa statistic (Ng, Lim, & Foo 1999; Carletta 1996; Siegel & Castellan 1988). It basically estimates the degree of agreement between two human subjects, but also considers the effect that the agreement is by chance. Suppose two human subjects are given n identical examples for labeling, and on a examples they give the same labels. The agreement rate between these two subjects is given by $P_a = a/n$. Assume that there are m different labels or classes in this task and c_j is the number of examples that both human subjects labeled as class j. The probability that these two annotators agree by chance is estimated by $P_e = \sum_{j=1}^{m}(\frac{c_j/2}{n})^2$. The Kappa statistic is then defined as: $\kappa = (P_a - P_e)/(1-P_e)$.

We judge the agreement of each pair of annotators using this method and report the averaged results in Table 1. The first half of the table lists the agreement rates on the original classification tasks, where the first row shows the Kappa values on the 4-scale labels and the second row shows the values on the reduced binary classes by treating Excellent and Good as positive and others as negative. The second half of the table shows the pairwise preference agreement. In this setting, we consider each pair of the suggestions s_i and s_j, along with the same query q. When both annotators ordered s_i and s_j equally, we say that they agree on the *preference* of these two suggestions. For example, if the labels given by one annotator are $L_1(q, s_i) =$ Excellent and $L_1(q, s_j) =$ Bad, and the labels given by the other are $L_2(q, s_i) =$ Good and $L_2(q, s_j) =$ Fair, these two annotators still agree on the preference, which is s_i is more similar to q than s_j, even though they give totally different labels. There are three preference choices: $s_i \rhd_j$, $s_i \lhd s_j$ and $s_i \equiv s_j$, which indicate that s_i is better, worse or equal to s_j. Similarly, this 3-scale label can be reduced to 2-scale by combining the equivalent case to either better or worse, which are denoted as $\unrhd$ vs. $\lhd$ and $\unlhd$ vs. $\rhd$, respectively.

From Table 1, we can see that the agreement on the 4-point scale labeling is unsurprisingly the lowest. Our annotators agree more with each other when reducing the 4 categories into a binary setting. The agreement on the 3-scale preference label is also higher than the original 4-point scale labeling, which motivates our comparison between direct and preference approaches to learning a similarity measure. However, when comparing the 2-scale preference label to the 2-point binary label, the distinctions between preference and classification are less clear.

Results

We compare the quality of several static similarity measures including the surface-matching, KL-divergence, Web-based kernel and our Web-relevance function, as well as the learned similarity functions based on different learning approaches. In particular, we concentrate on two evaluation metrics: the AUC score and precision at k.

The area under the ROC curve (AUC) has been used to indicate the quality of a ranking function. It essentially considers the correctness of the preference prediction of each pair of the elements in the sequence, and can be calculated by the following Wilcoxon-Mann-Whitney statistic (Cortes & Mohri 2004):

$$A(f; \mathbf{x}, \mathbf{y}) = \sum_{i,j: y_i > y_j} \mathbf{I}_{f(x_i) > f(x_j)} + \frac{1}{2} \mathbf{I}_{f(x_i) = f(x_j)},$$

where f is the ordinal function derived using either the similarity metric or preference predictor, $\mathbf{x}$ is the sequence of compared elements and $\mathbf{y}$ is the labels. When evaluating the similarity between each query suggestion $s \in \{s_1, \cdots, s_n\}$ and the original query q, the AUC score basically says that we get one point for a correct preference prediction and half a point when the similarity measure cannot distinguish the preference.

Another metric that is commonly used in a ranking scenario is the *precision at k*, which calculates the accuracy of the top-ranked k elements. Unlike the AUC score, which treats each pair of the elements in the sequence equally important, the precision at k metric only measures the quality of the top ranked items and ignores the rest.

When deriving the total ranking of the suggestions for each target query, if two suggestions are considered equally similar to the query, their order will be decided randomly for a fair comparison.

Measure	AUC	Prec@1	Prec@3	Prec@5	Cov.
Matching	0.617	0.633	0.444	0.368	64.7%
Dice Coefficient	0.627	0.708	0.456	0.383	64.7%
Jaccard Coefficient	0.627	0.708	0.456	0.383	64.7%
Overlap	0.606	0.442	0.389	0.352	64.7%
Cosine	0.626	0.708	0.456	0.373	64.7%
KL-Divergence	0.691	0.617	0.483	0.413	80.5%
Web-kernel	0.664	0.667	0.436	0.383	82.4%
Web-relevance	0.703	0.683	0.508	0.427	83.3%
Metric learning	0.735	0.717	0.556	0.477	94.4%
Preference learning	0.739	0.733	0.569	0.488	94.4%

Table 2: The AUC scores, precision at 1, 3, 5 and coverage of different similarity measures for short text segments

In addition to these two evaluation metrics, we also report the coverage of different similarity measures. For surface-matching methods, if the score is 0, which means there are no overlapping words in the two compared text segments, we treat this input pair as not covered. For corpus-based methods, if no expanded representation can be generated for any of the input text segments (i.e., no page that has the input text segment can be found by the search engine), then it is considered as not covered. Learning-base methods take several similarity measures as input features, so the only cases that they do no cover are those which are not covered by any of the underlaying static similarity measures. When comparing the performance of two methods, we conduct the statistical significance test in the following way. First, we group the query/suggestion pairs by the target queries. Evaluation metrics such as AUC or precision at k are computed for each of target queries. We then run a student's paired-t test on these individual scores and consider the results to be statistically significant when the p-value is lower than 0.05.

Table 2 lists the scores of the similarity measures we tested in different evaluation metrics. When evaluating static similarity measures, the average performance on all the labeled query/suggestion pairs is reported. For metric learning and preference learning, the experiments were conducted using 10-fold cross validation and the averaged performance on the 10 disjoint subsets for testing is reported.

Generally speaking, the surface-matching methods cover the least number of cases (64.7%). They also have worse total ranking results, which is reflected by their lower AUC scores compared to other groups of similarity measures. However, when only the top ranked suggestions are considered for evaluation, the surface-matching methods perform reasonably well, especially for the Dice coefficient, Jaccard coefficient and the cosine measure. This may be due to the fact that many of the candidate suggestions do have words that also occur in the corresponding query. In addition, when a pair of query and suggestion have no overlapping words, the chance that they are considered similar by our annotators is usually low, at least in this data set.

With the help of the Web, corpus-based methods cover more than 80% of the cases[3] and provide better overall rank-

[3]We ran these corpus-based methods at different time. Because

ing than the surface-matching methods. The KL-divergence method performs better than the Web-kernel approach in AUC, precision at 3 and precision 5, and worse on precision at 1. However, none of their differences is statistically significant. Our Web-relevance similarity measure is better than the original Web-kernel in all the evaluation metrics here. Except for precision at 1, all the differences are statistically significant. It is also better when compared to the KL-divergence method, although only the difference in the AUC score is statistically significant.

Taking the output of various similarity measures and combining them using machine learning have clear advantages. Both the AUC and precision scores indicate that the learning approach yields better quality in measuring short text segment similarity. Compared to the best corpus-based method, Web-relevance similarity measure, both learning approaches outperform it and all the differences except in precision at 1 are statistically significant. Although the preference learning approach has higher AUC and precision at k scores compared to the metric learning approach, the differences are in fact not statistically significant.

Conclusions

Web tasks such as query/keyword matching and search query suggestion rely heavily on the quality of similarity measures between short text segments. In this paper, we develop a Web-relevance similarity measure that naturally extends the recently proposed Web-based kernel function. Our method is not only better than a variety of surface-matching methods in almost all evaluation metrics in our experiments, but also outperforms existing state-of-the-art corpus-based approaches, such as KL-divergence and Web-kernel. We also demonstrate how to combine several similarity measures using machine learning. Our approach to learning takes the output of similarity measures including our Web-relevance similarity measure and other features to create a similarity measure that has broader coverage and also tunes the measure to a specific target application. We consider two approaches to learning (*similarity metric* and *preference ordering*) and both achieved higher AUC and precision scores as compared to all other similarity measures we tested using a query suggestion data set.

In the future, we plan to evaluate alternative approaches to learning the similarity of short text segments and alternative features, such as the use of ontology. As observed by (Mena *et al.* 2000), a term may have different senses or concepts and exist in heterogeneous ontogologies. Whether a pair of short text segments can be related to each other in the same ontology can be a useful feature. In addition, we would like to apply our similarity measures to different Web applications such as recommending advertising keywords for products in an online advertising scenario.

Acknowledgments

We thank Don Metzler, Sue Dumais and Chris Burges for helpful discussions. We are also grateful to anonymous reviewers for their valuable comments.

of the results returned by the search engine varied slightly, there is a small difference in the coverage of these methods.

References

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N.; and Hullender, G. 2005. Learning to rank using gradient descent. In *Proc. of ICML-05*, 89–96.

Carletta, J. 1996. Assessing agreement on classification tasks: the kappa static. *Computational Linguistics* 22(2):249–254.

Cortes, C., and Mohri, M. 2004. AUC optimization vs. error rate minimization. In *NIPS-03*.

Fitzpatrick, L., and Dent, M. 1997. Automatic feedback using past queries: Social searching? In *SIGIR*, 306–313.

Frank, E.; Paynter, G. W.; Witten, I. H.; Gutwin, C.; and Nevill-Manning, C. G. 1999. Domain-specific keyphrase extraction. In *Proc. of IJCAI-99*, 668–673.

Goodman, J., and Carvalho, V. R. 2005. Implicit queries for email. In *CEAS-05*.

Goodman, J. 2002. Sequential conditional generalized iterative scaling. In *ACL '02*.

Herbrich, R.; Graepel, T.; and Obermayer, K. 2000. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* 115–132.

Jones, R.; Rey, B.; Madani, O.; and Greiner, W. 2006. Generating query substitutions. In *Proc. of WWW '06*.

Landauer, T.; Foltz, P.; and Laham, D. 1998. An introduction to latent semantic analysis. *Discourse Processes* 25:259–284.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mena, E.; Kashyap, V.; Illarramendi, A.; and Sheth, A. P. 2000. Imprecise answers in distributed environments: Estimation of information loss for multi-ontology based query processing. *International Journal of Cooperative Information Systems* 9(4):403–425.

Metzler, D.; Dumais, S.; and Meek, C. 2007. Similarity measures for short segments of text. In *Proc. of ECIR-07*.

Ng, H.; Lim, C.; and Foo, S. 1999. A case study on inter-annotator agreement for word sense disambiguation. In *SIGLEX-99*, 9–13.

Sahami, M., and Heilman, T. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proc. of WWW '06*.

Siegel, S., and Castellan, N. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill.

Turney, P. D. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval* 2(4):303–336.

Turney, P. D. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proc. of ECML-01*.

Turney, P. D. 2003. Coherent keyphrase extraction via web mining. In *Proc. of IJCAI-03*, 434–439.

Yih, W.; Goodman, J.; and Carvalho, V. 2006. Finding advertising keywords on web pages. In *Proc. of WWW '06*.