A Web-based Kernel Function for Measuring the Similarity of Short Text Snippets

Mehran Sahami
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
sahami@google.com

Timothy D. Heilman
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043 USA
tdh@google.com

ABSTRACT

Determining the similarity of short text snippets, such as search queries, works poorly with traditional document similarity measures (e.g., cosine), since there are often few, if any, terms in common between two short text snippets. We address this problem by introducing a novel method for measuring the similarity between short text snippets (even those without any overlapping terms) by leveraging web search results to provide greater context for the short texts. In this paper, we define such a similarity kernel function, mathematically analyze some of its properties, and provide examples of its efficacy. We also show the use of this kernel function in a large-scale system for suggesting related queries to search engine users.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms

Algorithms, Experimentation

Keywords

Text similarity measures, Web search, Information retrieval, Kernel functions, Query suggestion

1. INTRODUCTION

In analyzing text, there are many situations in which we wish to determine how similar two short text snippets are. For example, there may be different ways to describe some concept or individual, such as "United Nations Secretary-General" and "Kofi Annan", and we would like to determine that there is a high degree of *semantic* similarity between these two text snippets. Similarly, the snippets "AI" and "Artificial Intelligence" are very similar with regard to their meaning, even though they may not share any actual terms in common.

Directly applying traditional document similarity measures, such as the widely used cosine coefficient [17, 16], to such short text snippets often produces inadequate results, however. Indeed, in both the examples given previously, applying the cosine would yield a similarity of 0 since each given text pair contains no common terms. Even in cases where two snippets may share terms, they may be using the term in different contexts. Consider the snippets "graphical models" and "graphical interface". The first uses *graphical* in reference to graph structures whereas the second uses the term to refer to graphic displays. Thus, while the cosine score between these two snippets would be 0.5 due to the shared lexical term "graphical", at a semantic level the use of this shared term is not truly an indication of similarity between the snippets.

To address this problem, we would like to have a method for measuring the similarity between such short text snippets that captures more of the semantic context of the snippets rather than simply measuring their term-wise similarity. To help us achieve this goal, we can leverage the large volume of documents on the web to determine greater context for a short text snippet. By examining documents that contain the text snippet terms we can discover other contextual terms that help to provide a greater context for the original snippet and potentially resolve ambiguity in the use of terms with multiple meanings.

Our approach to this problem is relatively simple, but surprisingly quite powerful. We simply treat each snippet as a query to a web search engine in order to find a number of documents that contain the terms in the original snippets. We then use these returned documents to create a *context vector* for the original snippet, where such a context vector contains many words that tend to occur in context with the original snippet (i.e., query) terms. Such context vectors can now be much more robustly compared with a measure such as the cosine to determine the similarity between the original text snippets. Furthermore, since the cosine is a valid kernel, using this function in conjunction with the generated context vectors makes this similarity function applicable in any kernel-based machine learning algorithm [4] where (short) text data is being processed.

While there are many cases where getting a robust measure of similarity between short texts is important, one particularly useful application in the context of search is to suggest related queries to a user. In such an application, a user who issues a query to a search engine may find it helpful to be provided with a list of semantically related queries that he or she may consider to further explore the related information space. By employing our short text similarity

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

kernel, we could match the user's initial query against a large repository of existing user queries to determine other similar queries to suggest to the user. Thus, the results of the similarity function can be directly employed in an end-user application.

The approach we take in constructing our similarity function has relations to previous work in both the Information Retrieval and Machine Learning communities. We explore these relations and put our work in the context of previous research in Section 2. We then formally define our similarity function in Section 3 and present initial examples of its use in Section 4. This is followed by a mathematical analysis of the similarity function in Section 5. Section 6 presents a system for related query suggestion using our similarity function, and an empirical evaluation of this system is given in Section 7. Finally, in Section 8 we provide some conclusions and directions for future work.

2. RELATED WORK

The similarity function we present here is based on *query expansion* techniques [3, 13] which have long been used in the Information Retrieval community. Such methods automatically augment a user query with additional terms based on documents that are retrieved in response to the initial user query or by using an available thesaurus. Our motivation for and usage of query expansion greatly differs from this previous work, however. First, the traditional goal of query expansion has been to improve recall (potentially at the expense of precision) in a retrieval task. Our focus, however, is on using such expansions to provide a richer representation for a short text in order to potentially compare it robustly with other short texts. Moreover, traditional expansion is focused on creating a new query for retrieval rather than doing pair-wise comparisons between short texts. Thus, the approach we take is quite different than the use of query expansion in a standard Information Retrieval context.

Alternatively, information retrieval researchers have previously proposed other means of determining query similarity. One early method proposed by Raghavan and Sever [14] attempts to measure the relatedness of two queries by determining differences in the ordering of documents retrieved in response to the two queries. This method requires a total ordering (ranking) of documents over the whole collection for each query. Thus, comparing the pairwise differences in rankings requires $O(N^2)$ time, where N is the number of documents in the collection. In the context of the web, where $N > 20$ billion[1], this algorithm quickly becomes intractable.

Later work by Fitzpatrick and Dent [9] measures query similarity using the normalized set overlap (intersection) of the top 200 documents retrieved for each query. While this algorithm's runtime complexity easily scales to the web, it will likely not lead to very meaningful similarity results as the sheer number of documents in the web collection will often make the set overlap for returned results extremely small (or empty) for many related queries that are not nearly identical. We show that this is indeed the case in our experimental and theoretical results later in the paper.

In the context of Machine Learning, there has been a great deal of work in using kernel methods, such as Support Vector Machines for text classification [11, 8]. Such work has recently extended to building specialized kernels aimed at measuring semantic similarity between documents. We outline some of these approaches below, and show how they differ from the work presented here.

One of the early approaches in this vein is *Latent Semantic Kernels* (LSK) [5], which is a kernel-based extension to the well-known Latent Semantic Indexing (LSI) [6] proposed in the Information Retrieval community. In LSK, a kernel matrix is computed over text documents, and the eigen-decomposition of this matrix is used to compute a new (lower rank approximation) basis for the space. The dimensions of the new basis can intuitively be thought of as capturing "semantic concepts" (i.e., roughly corresponding to co-varying subsets of the dimensions in the original space). While there may be some superficial similarities, this approach differs in fundamental respects from our work. First, our method is aimed at constructing a new kernel function, not using an existing kernel matrix to infer "semantic dimensions". Also, our method takes a *lazy* approach in the sense that we need not compute an expansion for a given text snippet until we want to evaluate the kernel function. We never need to explicitly compute a full kernel matrix over some set of existing text snippets nor its eigen-decomposition. Indeed, the kernel we present here is entire complimentary to work on LSK, as our kernel could be used to construct the kernel matrix on which the eigen-decomposition is performed.

An approach more akin to that taken here is the work of Kandola *et al.* [12] who define a kernel for determining the similarity of individual terms based on the collection of documents that these terms appear in. In their work, they learn a *Semantic Proximity Matrix* that captures the relatedness of individual terms by essentially measuring the correlation in the documents that contain these terms. In our work, the kernel we consider is not attempting to just determine similarity between single terms, but entire text snippets. Moreover, our approach does not require performing an optimization over an entire collection of documents (as is required in the previous work), but rather the kernel between snippets can be computed on-line selectively, as needed.

Previous research has also tried to address learning a semantic representation for a document by using cross-lingual techniques [18]. Here, one starts with a corpus of document pairs, where each pair is the same document written in two different languages. A correlation analysis is then performed between the corpora in each language to determine combinations of related words in one language that correlate well with combinations of words in the other language, and thereby learn word relations within a given language. Obviously, the approach we take does not require such paired corpora. And, again, we seek to not just learn relationships between single terms but between entire arbitrary short texts.

Thus, while there has been a good deal of work in determining semantic similarities between texts (which highlights the general importance of this problem), many of which use kernel methods, the approach we present has significant differences with that work. Moreover, our approach provides the compelling advantage that semantic similarity can be measured between multi-term short texts, where the entire text can be considered as a whole, rather than just determin-

[1] Leading search engines claim index sizes of at least 20 billion documents at the time of this writing.

ing similarity between individual terms. Furthermore, no expensive pre-processing of a corpus is required (e.g., eigen-decomposition), and the kernel can easily be computed for a given snippet pair as needed. We simply require access to a search engine (i.e., text index) over a corpus, which can be quite efficiently (linearly) constructed or can be obviated entirely by accessing a public search engine on the Web, such as the Google API (`http://www.google.com/apis`).

3. A NEW SIMILARITY FUNCTION

Presently, we formalize our kernel function for semantic similarity. Let x represent a short text snippet[2]. Now, we compute the *query expansion* of x, denoted $QE(x)$, as follows:

1. Issue x as a query to a search engine S.
2. Let $R(x)$ be the set of (at most) n retrieved documents $d_1, d_2, \ldots, d_n$
3. Compute the TFIDF term vector v_i for each document $d_i \in R(x)$
4. Truncate each vector v_i to include its m highest weighted terms
5. Let $C(x)$ be the centroid of the L_2 normalized vectors v_i:
$$C(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{v_i}{\|v_i\|_2}$$
6. Let $QE(x)$ be the L_2 normalization of the centroid $C(x)$:
$$QE(x) = \frac{C(x)}{\|C(x)\|_2}$$

We note that to be precise, the computation of $QE(x)$ really should be parameterized by both the query x and the search engine S used. Since we assume that S remains constant in all computations, we omit this parameter for brevity.

There are several modifications that can be made to the above procedure, as appropriate for different document collections. Foremost among these is the term weighting scheme used in Step 3. Here, we consider a TFIDF vector weighting scheme [15], where the weight $w_{i,j}$ associated with with term t_i in document d_j is defined to be:

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_i}),$$

where $tf_{i,j}$ is the frequency of t_i in d_j, N is the total number of documents in the corpus, and df_i is the total number of documents that contain t_i. We compute N and df_i using a large sample of documents from the web. Clearly, other weighting schemes are possible, but we choose TFIDF here since it is commonly used in the IR community and we have found it to empirically give good results in building representative query expansions. Also, in Step 4, we set the maximum number of terms in each vector $m = 50$, as we have found this value to give a good trade-off between representational robustness and efficiency.

Also, in Step 2, we need not choose to use the entirety of retrieved documents in order to produce vectors. We may choose to limit ourselves to create vectors using just the *contextually descriptive text snippet* for each document that is commonly generated by Web search engines. This would make our algorithm more efficient in terms of the amount of data processed, and allows us to make ready use of the results from public web search engines without having to even retrieve the full actual underlying documents. Of course, there remains the question of how large such descriptive texts provided by search engines need to be in order to be particularly useful. Empirically, we have found that using 1000 characters (in a token delimited window centered on the original query terms in the original text) is sufficient to get accurate results, and increasing this number does not seem to provide much additional benefit.

Evaluating a variety of term weighting or text windowing schemes, however, is not the aim of this work and we do not explore it further here. Rather we simply seek to outline some of the issues that may be of interest to practitioners and provide some guidance on reasonable values to use that we have found work well empirically.

Finally, given that we have a means for computing the query expansion for a short text, it is a simple matter to define the semantic kernel function K as the inner product of the query expansions for two text snippets. More formally, given two short text snippets x and y, we define the semantic similarity kernel between them as:

$$K(x, y) = QE(x) \cdot QE(y).$$

Observation 1. *$K(x, y)$ is a valid kernel function.*

This readily follows from the fact that $K(x, y)$ is defined as an inner product with a bounded norm (given that each query expansion vector has norm 1.0). For more background on the properties of kernel functions and some of their potential applications, we refer the interested reader to the text by Cristianini and Shawe-Taylor [4].

4. INITIAL RESULTS WITH KERNEL

To get a cursory evaluation for how well our semantic similarity kernel performs, we show results with the kernel on a number of text pairs, using the Google search engine as the underlying document retrieval mechanism. We attempt to highlight both the strengths and potential weaknesses of this kernel function.

We examined several text snippet pairs to determine the similarity score given by our new web-based kernel, the traditional cosine measure, and the set overlap measure proposed by Fitzpatrick and Dent. We specifically look at three genres of text snippet matching: (i) acronyms, (ii) individuals and their positions, and (iii) multi-faceted terms.[3] Examples of applying the kernel are shown in Table 1, which is segmented by the genre of matching examined.

[2] While the real focus of our work is geared toward short text snippets, there is no technical reason why x must have limited length, and in fact x can be arbitrary text.

[3] We prefer the term *multi-faceted* over *ambiguous*, since multi-faceted terms may have the same definition in two contexts, but the accepted semantics of that definition may vary in context. For example, the term "travel" has the same definition in both the phrases "space travel" and "vacation travel", so it is (strictly speaking) not ambiguous here, but the semantics of what is meant by *traveling* in those two cases is different.

Text 1	Text 2	Kernel	Cosine	Set Overlap
Acronyms				
support vector machine	SVM	0.812	0.0	0.110
portable document format	PDF	0.732	0.0	0.060
artificial intelligence	AI	0.831	0.0	0.255
artificial insemination	AI	0.391	0.0	0.000
term frequency inverse document frequency	tf idf	0.831	0.0	0.125
term frequency inverse document frequency	tfidf	0.507	0.0	0.060
Individuals and their positions				
UN Secretary-General	Kofi Annan	0.825	0.0	0.065
UN Secretary-General	George W. Bush	0.110	0.0	0.000
US President	George W. Bush	0.688	0.0	0.045
Microsoft CEO	Steve Ballmer	0.838	0.0	0.090
Microsoft CEO	Bill Gates	0.317	0.0	0.000
Microsoft Founder	Bill Gates	0.677	0.0	0.010
Google CEO	Eric Schmidt	0.845	0.0	0.105
Google CEO	Larry Page	0.450	0.0	0.040
Google Founder	Larry Page	0.770	0.0	0.050
Microsoft Founder	Larry Page	0.189	0.0	0.000
Google Founder	Bill Gates	0.096	0.0	0.000
web page	Larry Page	0.123	0.5	0.000
Multi-faceted terms				
space exploration	NASA	0.691	0.0	0.070
space exploration	space travel	0.592	0.5	0.005
vacation travel	space travel	0.321	0.5	0.000
machine learning	ICML	0.586	0.0	0.065
machine learning	machine tooling	0.197	0.5	0.000
graphical UI	graphical models	0.275	0.5	0.000
graphical UI	graphical interface	0.643	0.5	0.000
java island	Indonesia	0.454	0.0	0.000
java programming	Indonesia	0.020	0.0	0.000
java programming	applet development	0.563	0.0	0.010
java island	java programming	0.280	0.5	0.000

Table 1: Examples of web-based kernel applied to short text snippet pairs.

The first section of the table deals with the identification of acronyms. In this genre, we find two notable effects using our kernel. First, from the relatively high similarity scores found between acronyms and their full name, it appears that our kernel is generally effective at capturing the semantic similarity between an acronym and its full name. Note that the kernel scores are not 1.0 since acronyms can often have multiple meanings. Related to this point, our second observation is that our kernel function (being based on contextual text usage on the web) tends to prefer more common usages of an acronym in determining semantic similarity. For example, the text "AI" is determined to be much more similar to "artificial intelligence" than "artificial insemination" (even though it is a valid acronym for both), since contextual usage of "AI" on the web tends to favor the former meaning. We see a similar effect when comparing "term frequency inverse document frequency" to "tf idf" and "tfidf". While the former acronym tends to be more commonly used (especially since the sub-acronyms "tf" and "idf" are separated), the still reasonable score over 0.5 for the acronym "tfidf" shows that the kernel function is still able to determine a solid level of semantic similarity. It is not surprising that the use of cosine similarity is entirely inappropriate for such a task (since the full name of an acronym virtually never contains the acronym itself). Moreover, we find, as expected, that the set overlap measure leads to very low (and not very robust) similarity values.

Next, we examined the use of our kernel in identifying different characterizations of individuals. Specifically, we considered determining the similarity of the name of a notable individual with his prominent role description. The results of these examples are shown in the second section of Table 1.

In order to assess the strengths and weaknesses of the kernel function we intentionally applied the kernel to both correct pairs of descriptions and individuals as well looking at pairs involving an individual and a *close*, but incorrect, description. For example, while Kofi Annan and George W. Bush are both prominent world political figures, the kernel is effective at determining the correct role matches and assigning them appropriately high scores.

In the realm of business figures, we find that the kernel is able to distinguish Steve Ballmer as the current CEO of Microsoft (and not Bill Gates). Bill Gates still gets a non-trivial semantic similarity with the role "Microsoft CEO" since he was indeed the *former* CEO, but he is much more strongly (by a over a factor of 2) associated correctly with the text "Microsoft founder". Similarly, the kernel is successful at correctly identifying the current Google CEO (Eric Schmidt) from Larry Page (Google's founder and *former* CEO).

We also attempted to test how readily the kernel function assigned high scores for inappropriate matches by trying to pair Bill Gates as the founder of Google and Larry Page as the founder of Microsoft. The low similarity scores given by the kernel show that it does indeed find little semantic similarity between these inappropriate pairs. Once again, the kernel value is non-zero since each of the individuals is indeed the founder of *some* company, so the texts compared are not entirely devoid of some semantic similarity. Finally, we show that even though Larry Page has a very common surname, the kernel does a good job of not confusing him with a "web page" (although the cosine gives a inappropriately high similarity due to the match on the term "page").

Lastly, we examined the efficacy of the kernel when applied to texts with multi-faceted terms – a case where we expect the raw cosine and set overlap to once again do quite poorly. As expected, the kernel does a reasonable job of determining the different facets of terms, such as identifying "space exploration" with "NASA" (even though they share no tokens), but finding that the similarity between "vacation travel" and "space travel" is indeed less than the cosine might otherwise lead us to believe. Similar effects are seen in looking at terms used in context, such as "machine", "graphical", and "java". We note that in many cases, the similarity values here are not as extreme as in the previous instances. This has to do with the fact that we are trying to measure the rather fuzzy notion of *aboutness* between semantic concepts rather than trying to identify an acronym or individual (which tend to be much more specific matches). Still, the kernel does a respectable job (in most cases) of providing a score above 0.5 when two concepts are very related and less than 0.3 when the concepts are generally thought of as distinct.

Once again, the low similarity scores given by the set overlap method show that in the context of a large document collection such as the web, this measure is not very robust. As a side note, we also measured the set overlap using the top 500 and top 1000 documents retrieved for each query (in addition to the results reported here which looked at the top 200 documents as suggested in the original paper), and found qualitatively very similar results thus indicating that the method itself, and not merely the parameter settings, led to the poor results in the context of the web.

5. THEORETICAL ANALYSIS OF KERNEL AND SET OVERLAP MEASURES

In light of the anecdotal results in comparing our kernel function with the set overlap measure, it is useful to mathematically analyze the behavior of each measure in the context of large (and continuously growing) document collections such as the web. We begin by introducing some relevant concepts for this analysis.

Definition 1. Two documents are *ϵ-indistinguishable* to a search engine S with respect to a query q if the search engine finds both documents to be equally relevant to the query within the tolerance ϵ of its ranking function.

Intuitively, this definition captures the notion that since a search engine generates a ranking of documents by scoring them according to various criteria, the scores used for ranking may only accurately resolve document relevance to within some toleration ϵ. This ϵ toleration factor reflects the inherent resolving limitation of a given relevance scoring function, and thus within this toleration factor, the ranking of documents can be seen as arbitrary.

As we are interested in analyzing very large corpora and the behavior of the various similarity measures in the limit as the collections being searched grow infinitely large, we consider the situation in which so many relevant documents are available to a search engine for any given query q that the set of n top-ranked documents $R(q)$ are all ϵ-indistinguishable. To formalize this concept, let $T_S(q)$ be the set of all (maximally ranked) documents which are all ϵ-indistinguishable to search engine S for query q. Now we note that as the size of the collection D grows to infinity (i.e., $|D| \rightarrow \infty$) then $|T_S(q)| \rightarrow \infty$, since there will be infinitely many documents that are equally relevant to a given query. Moreover, since the documents in $T_S(q)$ are ϵ-indistinguishably relevant to q, we assume that the top n results retrieved for query q will be a uniformly random sampled subset of $T_S(q)$ (with replacement, just to simplify the analysis in the limit as $T_S(q)$ grows large). The use of a uniform distribution for sampling documents from $T_S(q)$ can be justified by the fact that since all documents in $T_S(q)$ are within the tolerance ϵ of the ranking function, their ranking is arbitrary. Since in this context there is no reason to prefer one particular distribution of rankings over any another, a maximally entropic distribution (i.e., uniform) is a reasonable model to use.

In the sequel, assume that we are given two different queries q_1 and q_2, which are so highly related to each other that (again, for simplicity) we assume $T_S(q_1) = T_S(q_2)$. While in reality it is unlikely that two queries would share *exactly* the same set of maximally relevant documents, we make this assumption (which intuitively should lead to a very high similarity score between q_1 and q_2) to show that even under conditions of extreme similarity, there are shortcomings with the set overlap similarity measure. We show that the kernel function does not suffer from similar problems. Since we assume $T_S(q_1) = T_S(q_2)$ and always use the same search engine S in our analysis, we will simply refer to $T_S(q_1)$ (and thus $T_S(q_2)$) as T for brevity when there is no possibility of ambiguity.

5.1 Properties of Set Overlap measure

THEOREM 1. *Let $R(q)$ be the set of n top-ranked documents with respect to query q. Then, in the set overlap measure, the expected normalized set overlap for queries q_1 and q_2, is*

$$\frac{1}{n}E(|R(q_1) \cap R(q_2)|) = \frac{n}{|T|}.$$

PROOF. This follows from the fact that a results set $R(q_1)$ of size n for query q_1 contains $\frac{n}{|T|}$ of the documents in T. When we then uniformly sample n documents from T to produce the results set $R(q_2)$ for query q_2, our probability of picking a document in $R(q_1)$ on each draw is simply $\frac{n}{|T|}$. Thus, after n draws, the expected overlap $E(|R(q_1) \cap R(q_2)|) = \frac{n^2}{|T|}$, and normalizing this value by the number of draws yields the desired result: $\frac{1}{n}E(|R(q_1) \cap R(q_2)|) = \frac{n}{|T|}$. □

A desirable (and straightforward) corollary of this theorem, is that as we increase the results set size to capture all the relevant documents (i.e., $n \to |T|$), the expected overlap measure approaches 1. Interestingly, however, for any fixed results set size n, as $|T| \to \infty$, the expected normalized set overlap $\frac{1}{n}E(|R(q_1) \cap R(q_2)|) \to 0$. This result suggests that even if two queries are so similar as to have the same set of highly relevant documents, in the limit as the collection size increases (and thus the number of relevant documents increases), the similarity as given by the set overlap measure will go to 0. Note that this problem would be even worse if we had not made the simplifying assumption that $T_S(q_1) = T_S(q_2)$, as the set overlap measure would approach 0 even more quickly. While this result is not surprising, it does show the problem that arises with using such a measure in the context of a large collection such as the web. This is also borne out in the anecdotal results seen in Section 4.

5.2 Properties of Kernel function

Analyzing our kernel function under the same conditions as above, we find that the measure is much more robust to growth in collection size, making the measure much more amenable for use in broad contexts such as the web. Since the kernel function computes vectors based on the documents retrieved from the relevant set T, we examine properties of the document vectors from this set. Namely, we assume that the document vectors v generated from the documents in T are distributed according to some arbitrary[4] distribution π with mean direction vector μ and a standard deviation σ, where σ measures the *angular* difference from μ. Such distributions, which are defined based on direction or angle, fall into the general class of *circular* distributions and a full discussion of them is beyond the scope of this paper (we refer the interested reader to work on document analysis using circular distributions, such as the von Mises distribution [7, 2].)

In this context, we note that $QE(q)$ for a given query q is simply a sample mean from the distribution π of document vectors in T. This follows from the fact that the set of relevant documents $R(q)$ retrieved in response to q are simply samples from T, and thus their corresponding document vectors v_i are just samples from π. The centroid of these vectors $C(q)$ is defined to be the mean (direction) of the vectors, and $QE(q)$ is just the unit length normalized centroid (with the same direction as $C(q)$) thus making it a sample mean of the vector directions in π.

Observation 2. *As $n \to |T|$, then $QE(q) \to \mu$.*

This observation follows directly from the fact that as $n \to |T|$, then the sample on which $QE(q)$ is based becomes the whole population, so $QE(q)$ becomes the true population mean μ.

Observation 3. *If queries q_1 and q_2 share the same ϵ-indistinguishable relevant set T, then as $n \to |T|$, it follows that $K(q_1, q_2) \to 1$.*

To show this observation we note that if q_1 and q_2 share the same ϵ-indistinguishable relevant set T, as $n \to |T|$, then $QE(q_1) \to \mu$ and $QE(q_2) \to \mu$. Thus, $K(q_1, q_2) \to \mu \cdot \mu = 1$.

This gives us the intuitively desirable behavior (which is also shared with the set overlap measure) that as the size of the results set used to generate the query expansion vectors grows to encompass all relevant documents, the similarity for two queries with the same results set goes to 1.

In contrast to the set overlap measure, we find that the kernel function does not go to 0 as the number of documents in the relevant results set T increases without bound. Indeed, we can prove a stronger theorem with respect to the property of our kernel function in the limit.

Theorem 2. *Let σ be the standard deviation of the distribution π of vectors corresponding to documents in the ϵ-indistinguishable set of query results T for queries q_1 and q_2. Then, with high probability (> 98%), it holds that*

$$\cos^{-1} K(q_1, q_2) \leq 5.16 \frac{\sigma}{\sqrt{n}}.$$

Proof. To prove this result, we begin by noting that $QE(q)$ is simply a sample from the sampling distribution (hereafter denoted ψ_μ) for the mean μ of π. Thus, by the Central Limit Theorem, the distribution ψ_μ is approximately normal with mean μ and standard deviation $\frac{\sigma}{\sqrt{n}}$ regardless of the shape of the original vector distribution π. Now, let $\theta_{1,\mu}$ be the angle between $QE(q_1)$ and μ, and similarly, let $\theta_{2,\mu}$ be the angle between $QE(q_2)$ and μ. Leveraging the approximate normality of ψ_μ, with 99% probability it follows that $\theta_{1,\mu} \leq 2.58\frac{\sigma}{\sqrt{n}}$ and similarly with 99% probability we have $\theta_{2,\mu} \leq 2.58\frac{\sigma}{\sqrt{n}}$. Thus, combining these results using the Union Bound we have that, with 98% probability,

$$\theta_{1,\mu} + \theta_{2,\mu} \leq 5.16 \frac{\sigma}{\sqrt{n}}. \quad (1)$$

Let $\theta_{1,2}$ denote the angle between $QE(q_1)$ and $QE(q_2)$. By the triangle inequality for angles, it holds that $\theta_{1,2} \leq \theta_{1,\mu} + \theta_{2,\mu}$. Substituting into Equation 1, yields

$$\theta_{1,2} \leq 5.16 \frac{\sigma}{\sqrt{n}}. \quad (2)$$

Now, noting that $K(q_1, q_2) = \cos\theta_{1,2}$ (and applying $\cos^{-1}$ is well-defined here since $\theta_{1,2}$ is always in the interval [0, $\frac{\pi}{2}$] given that all document vectors only have non-negative component values), we obtain the desired result:

$$\cos^{-1} K(q_1, q_2) = \theta_{1,2} \leq 5.16 \frac{\sigma}{\sqrt{n}}. \quad (3)$$

□

Note that the bound on $\cos^{-1} K(q_1, q_2)$ in Theorem 2 is independent of $|T|$, even though it depends on σ. This follows from the fact that since the vectors that correspond to documents in T are just samples from some true underlying stationary distribution π (with mean μ and standard deviation σ), the true standard deviation σ does not change as $|T| \to \infty$. Since $\cos^{-1} K(q_1, q_2)$ is independent of $|T|$, then so is $K(q_1, q_2)$. This implies that the kernel function is robust for use in large collections, as its value does not depend on the number of relevant documents, but simply on the directional dispersion (measured by the standard deviation over angles) of the vectors of the relevant documents. This property makes the kernel well-suited for use with large collections such as the web.

[4] The distribution π is arbitrary up to the fact that its first two moments, mean and variance, exist (which is a fairly standard and non-restrictive assumption).

Furthermore, we can consider the more general (and realistic) case where the sets of ϵ-indistinguishable results for queries q_1 and q_2 need not be the same (i.e., $T_S(q_1) \neq T_S(q_2)$), and now prove a more general result that subsumes Theorem 2 as a special case.

THEOREM 3. *Let μ_1 and μ_2 be the respective means of the distributions π_1 and π_2 of vectors corresponding to documents from $T_S(q_1)$ and $T_S(q_2)$. Let σ_1 and σ_2 be the standard deviations of π_1 and π_2, respectively. And let θ_{μ_1,μ_2} be the angle between μ_1 and μ_2. Then, with high probability (> 98%), it holds that*

$$\cos^{-1} K(q_1, q_2) \leq 2.58 \frac{\sigma_1 + \sigma_2}{\sqrt{n}} + \theta_{\mu_1,\mu_2}.$$

PROOF. We prove this result in a similar manner to Theorem 2. First, we define θ_{1,μ_1} as the angle between $QE(q_1)$ and μ_1, and θ_{2,μ_2} as the angle between $QE(q_2)$ and μ_2. As before, we note that $QE(q_1)$ and $QE(q_2)$ are simply respective samples from the sampling distributions (denoted ψ_{μ_1} and ψ_{μ_2}) for the means μ_1 of π_1 and μ_2 of π_2. Once again invoking the Central Limit Theorem, we know that ψ_{μ_1} and ψ_{μ_2} are approximately normal, and thus:

$$\theta_{1,\mu_1} \leq 2.58 \frac{\sigma_1}{\sqrt{n}} \text{ with 99\% probability,} \tag{4}$$

and

$$\theta_{2,\mu_2} \leq 2.58 \frac{\sigma_2}{\sqrt{n}} \text{ with 99\% probability.} \tag{5}$$

Combining Equations 4 and 5 using the Union Bound yields that with 98% probability:

$$\theta_{1,\mu_1} + \theta_{2,\mu_2} \leq 2.58 \frac{\sigma_1 + \sigma_2}{\sqrt{n}} \tag{6}$$

Adding θ_{μ_1,μ_2} to both sides of Equation 6 we obtain

$$\theta_{1,\mu_1} + \theta_{2,\mu_2} + \theta_{\mu_1,\mu_2} \leq 2.58 \frac{\sigma_1 + \sigma_2}{\sqrt{n}} + \theta_{\mu_1,\mu_2}. \tag{7}$$

By the triangle inequality for angles: $\theta_{2,\mu_1} \leq \theta_{2,\mu_2} + \theta_{\mu_1,\mu_2}$. Substituting in the equation above yields

$$\theta_{1,\mu_1} + \theta_{2,\mu_1} \leq 2.58 \frac{\sigma_1 + \sigma_2}{\sqrt{n}} + \theta_{\mu_1,\mu_2}. \tag{8}$$

Again, by the triangle inequality for angles we know that $\theta_{1,2} \leq \theta_{1,\mu_1} + \theta_{2,\mu_1}$, and substitution give us

$$\theta_{1,2} \leq 2.58 \frac{\sigma_1 + \sigma_2}{\sqrt{n}} + \theta_{\mu_1,\mu_2}. \tag{9}$$

As noted previously, we have $\theta_{1,2} = \cos^{-1} K(q_1, q_2)$, which combined with Equation 9 above gives us the desired result:

$$\cos^{-1} K(q_1, q_2) \leq 2.58 \frac{\sigma_1 + \sigma_2}{\sqrt{n}} + \theta_{\mu_1,\mu_2}. \tag{10}$$

□

We note that Theorem 2 is simply a special case of Theorem 3, where $\sigma_1 = \sigma_2$, $\mu_1 = \mu_2$, and thus $\theta_{\mu_1,\mu_2} = 0$. Theorem 2 was derived separately just to provide a direct contrast with the normalized set overlap measure under the same conditions.

Intuitively, Theorem 3 tells us that that the kernel function is essentially trying to measure the cosine of the angle θ_{μ_1,μ_2} between the mean vectors of the documents relevant to each query, with a "noise" term (proportional to $\frac{\sigma_1+\sigma_2}{\sqrt{n}}$) that depends on the natural dispersion (standard deviation) of the documents relevant to each query and the size n of the sample used to generate the query expansion. Thus, if we were to think of the set of documents that are relevant to a given query q as the "topic" of q, then the kernel is attempting to measure the mean "topical" difference between the queries, independent of the number of documents that make up each topic. This sort of behavior (and its independence from the overall collection size) is an intuitively desirable property for a similarity function.

6. RELATED QUERY SUGGESTION

Armed with promising anecdotal evidence as well as theoretical results that argue in favor of using this kernel when comparing short texts, we turn our attention to the task of developing a simple application based on this kernel. The application we choose is query suggestion—that is, to suggest potentially related queries to the users of a search engine to give them additional options for information finding. We note that there is a long history of work in query refinement, including the previously mentioned work in query expansion [3, 13], harnessing relevance feedback for query modification [10], using pre-computed term similarities for suggestions [19], linguistically mining documents retrieved in response to a search for related terms and phrases [20, 1], and even simply finding related queries in a thesaurus. While this is certainly an active area of work in information retrieval, we note that improving query suggestion is not the primary focus of this work. Thus, we intentionally do not compare our system with others. Rather, we use query suggestion as a means of showing the potential utility of our kernel function in just one, of potentially many, real-world applications. We provide a user evaluation of the results in this application to get a more objective measure of the efficacy of our kernel.

At a high-level, our query expansion system can be described as starting with an initial repository Q of previously issued user queries (for example, culled from search engine logs). Now, for any newly issued user query u, we can compute our kernel function $K(u, q_i)$ for all $q_i \in Q$ and suggest related queries q_i which have the highest kernel score with u (subject to some post-filtering to eliminate related queries that are too linguistically similar to each other).

More specifically, we begin by pre-computing the query expansions for a repository Q of approximately 116 million popular user queries issued in 2003, determined by sampling anonymized web search logs from the Google search engine. After generating these query expansions, we index the resulting vectors for fast retrieval in a retrieval system R. Now, for any newly observed user query u, we can generate its query expansion $QE(u)$ and use this entire expansion as a disjunctive query to R, finding all existing query expansions $QE(q_i)$ in the repository that potentially match $QE(u)$. Note that if a query expansion $QE(q)$ indexed in R does not match $QE(u)$ in at least one term (i.e., it is not retrieved), then we know $K(u, q) = 0$ since there are no common terms in $QE(u)$ and $QE(q)$. For each retrieved query expansion $QE(q_i)$, we can then compute the inner product $QE(u) \cdot QE(q_i) = K(u, q_i)$.

To actually determine which of the matched queries from

Original Query	Suggested Queries	Kernel Score	Human Rating
california lottery	california lotto home	0.812	3
	winning lotto numbers in california	0.792	5
	california lottery super lotto plus	0.778	3
valentines day	2003 valentine's day	0.832	3
	valentine day card	0.822	4
	valentines day greeting cards	0.758	4
	I love you valentine	0.736	2
	new valentine one	0.671	1

Table 2: Examples of suggested queries, along with corresponding kernel scores and human rater evaluations.

the repository to suggest to the user, we use the following algorithm, where the constant MAX is set to the maximum number of suggestions that we would like to obtain:

```
Given:   user query $u$, and
         list of matched queries from repository
Output:  list $Z$ of queries to suggest
1.      Initialize suggestion list $Z = \emptyset$
2.      Sort kernel scores $K(u, q_i)$ in descending order
        to produce an ordered list $L = (q_1, q_2, \ldots, q_k)$
        of corresponding queries $q_i$.
3.      $j = 1$
4.      While ($j \leq k$ and size($Z$) $<$ MAX) do
4.1         If ($|q_j| - |q_j \cap z| > 0.5|z| \quad \forall z \in (Z \cup \{u\})$) then
4.1.1           $Z = Z \cup \{q_j\}$
4.2         $j = j + 1$
5.      Return suggestion list $Z$
```

Here $|q|$ denotes the number of terms in query q. Thus, the test in Step 4.1 above is our post-filter to only add another suggested query q_j if it differs by more than half as many terms from any other query already in the suggestion list Z (as well as the original user query u). This helps promote linguistic diversity in the set of suggested queries. The outputted list of query suggestions Z can be presented to the search engine user to guide them in conducting follow-up searches.

7. EVALUATION OF QUERY SUGGESTION SYSTEM

In order to evaluate our kernel within the context of this query suggestion system, we enlisted nine human raters who are computer scientists familiar with information retrieval technologies. Each rater was asked to issue queries from the Google Zeitgeist[5] in a different month of 2003 (since our initial repository of queries to suggest was culled near the start of 2003). The Google Zeitgeist tracks popular queries on the web monthly. We chose to use such common queries for evaluation because if useful suggestions were found, they could potentially be applicable for a large number of search engine users who had the same information needs.

Each rater evaluated the suggested queries provided by the system on a 5-point Likert scale, defined as:

1: suggestion is totally off topic.

2: suggestion is not as good as original query.

3: suggestion is basically same as original query.

4: suggestion is potentially better than original query.

5: suggestion is fantastic – should suggest this query since it might help a user find what they're looking for if they issued it instead of the original query.

In our experiment we set the maximum number of suggestions for each query (MAX) to 5, although some queries yielded fewer than this number of suggestions due to having fewer suggestions pass the post-filtering process. A total of 118 user queries, which yielded 379 suggested queries (an average of 3.2 suggestions per query) were rated. Note that some raters evaluated a different number of queries than other raters.

In Table 2 we provide an example of two user queries, the query suggestions made using our system, the corresponding kernel scores, and the human evaluation ratings for the suggested queries. As can be seen in the first example, it is not surprising that users interested in the "california lottery" would prefer to find winning numbers rather than simply trying to get more information on the lottery in general. In the second example, we find that users querying for "valentines day" may be looking to actually send greeting cards. The suggestion "new valentine one" is actually referring to a radar detector named *Valentine One* and thus is clearly off-topic with regard to the original user query.

Since each query suggestion has a kernel score associated with it, we can determine how suggestion quality is correlated with the kernel score by looking at the average rating over all suggestions that had a kernel score above a given threshold. If the kernel is effective, we would generally expect higher kernel scores to lead to more useful queries suggested to the user (as they would tend to be more on-topic even given the post-filtering mechanism that attempts to promote diversity among the query suggestions). Moreover, we would expect that overall the suggestions would often be rated close to 3 (or higher) if the kernel were effective at identifying query suggestions semantically similar to the original query.

The results of this experiment are shown in Figure 1, which shows the average user rating for query suggestions, where we use a kernel score threshold to only consider sug-

[5] `http://www.google.com/intl/en/press/zeitgeist.html`

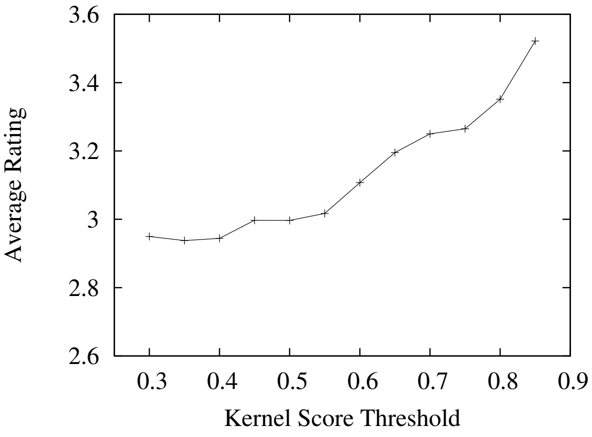

Figure 1: Average ratings at various kernel thresholds.

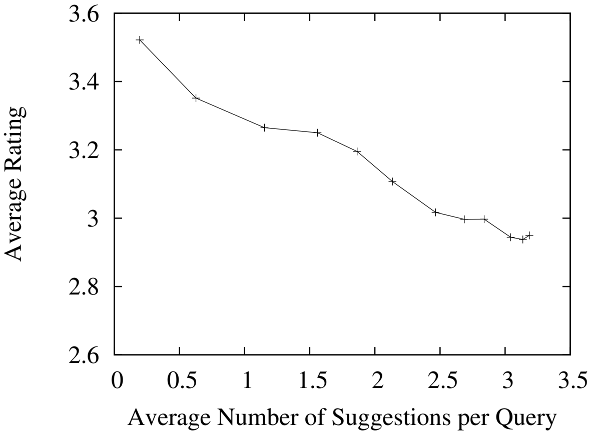

Figure 2: Average ratings versus average number of query suggestions made for each query as kernel threshold is varied from 0.85 down to 0.3.

gestions that scored at that threshold or higher with the original query. Indeed, we see that the query suggestions are generally rated close to 3 (same as the original query), but that the rating tends to increase with the kernel score. This indicates that queries deemed by the kernel to be very related to the original query are quite useful to users in honing their information need, especially when we allow for some diversity in the results using the post-filtering mechanism. In fact, we found that without the use of the post-filtering mechanism, the results suggested by the system were often too similar to the original query to provide much additional utility for query suggestion (although it was indicative of the kernel being effective at finding related queries).

Figure 2 shows a graph analogous to a Precision-Recall curve, where we plot the average user rating for query suggestions versus the average number of suggestions that are given per query as we vary the kernel score threshold from 0.85 down to 0.3. We see a clear trade-off between the quality of the suggestions presented to the user and the number of suggestions given. Indeed, it is possible, on average to give two query suggestions for each query which have a quality (slightly) higher than the original query.

8. CONCLUSIONS AND FUTURE WORK

We have presented a new kernel function for measuring the semantic similarity between pairs of short text snippets. We have shown, both anecdotally and in a human-evaluated query suggestion system, that this kernel is an effective measure of similarity for short texts, and works well even when the short texts being considered have no common terms. Moreover, we have also provided a theoretical analysis of the kernel function that shows that it is well-suited for use with the web.

There are several lines of future work that this kernel lays the foundation for. The first is improvement in the generation of query expansions with the goal of improving the match score for the kernel function. The second is the incorporation of this kernel into other kernel-based machine learning methods to determine its ability to provide improvement in tasks such as classification and clustering of text.

Also, there are certainly other potential web-based applications, besides query suggestion, that could be considered as well. One such application is in a question answering system, where the question could be matched against a list of candidate answers to determine which is the most similar semantically. For example, using our kernel we find that: K(“Who shot Abraham Lincoln”, “John Wilkes Booth”) = 0.730. Thus, the kernel does well in giving a high score to the correct answer to the question, even though it shares no terms in common with the question. Alternatively, K(“Who shot Abraham Lincoln”, “Abraham Lincoln”) = 0.597, indicating that while the question is certainly semantically related to “Abraham Lincoln”, the true answer to the question is in fact more semantically related to the question. Finally, we note that this kernel is not limited to use on the web, and can also be computed using query expansions generated over domain-specific corpora in order to better capture contextual semantics in particular domains. We hope to explore such research venues in the future.

Acknowledgments

We thank Amit Singhal for many invaluable discussions related to this research. Additionally, we appreciate the feedback provided on this work by the members of the Google Research group, especially Vibhu Mittal, Jay Ponte, and Yoram Singer. We are also indebted to the nine human raters who took part in the query suggestion evaluation.

9. REFERENCES

[1] P. Anick and S. Tipirneni. The paraphrase search assistant: Terminological feedback for iterative information seeking. In *Proceedings of the 22nd Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 153–159, 1999.

[2] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

[3] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *The Third Text REtrieval Conference*, pages 69–80, 1994.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[5] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2):127–152, 2002.

[6] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] I. S. Dhillon and S. Sra. Modeling data using directional distributions, 2003.

[8] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM-98: Proceedings of the Seventh International Conference on Information and Knowledge Management*, 1998.

[9] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: Social searching? In *Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313, 1997.

[10] D. Harman. Relevance feedback and other query modification techniques. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 241–263. Prentice Hall, 1992.

[11] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, 1998.

[12] J. S. Kandola, J. Shawe-Taylor, and N. Cristianini. Learning semantic similarity. In *Advances in Neural Information Processing Systems (NIPS) 15*, pages 657–664, 2002.

[13] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, 1998.

[14] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *Proceedings of the 18th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 344–350, 1995.

[15] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[16] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

[17] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

[18] A. Vinokourov, J. Shawe-Taylor, and N. Cristianini. Inferring a semantic representation of text via cross-language correlation analysis. In *Advances in Neural Information Processing Systems (NIPS) 15*, pages 1473–1480, 2002.

[19] B. Vlez, R. Wiess, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. In *Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 6–15, 1997.

[20] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.