Query by Document

Yin Yang
Computer Science
HKUST
yini@cs.ust.hk

Nilesh Bansal
Computer Science
University of Toronto
nilesh@cs.toronto.edu

Wisam Dakka
Computer Science
Columbia University
wisam@cs.columbia.edu

Panagiotis Ipeirotis
Information Systems
New York University
panos@stern.nyu.edu

Nick Koudas
Computer Science
University of Toronto
koudas@cs.toronto.edu

Dimitris Papadias
Computer Science
HKUST
dimitris@cs.ust.hk

ABSTRACT

We are experiencing an unprecedented increase of content contributed by users in forums such as blogs, social networking sites and micro-blogging services. Such abundance of content complements content on web sites and traditional media forums such as news papers, news and financial streams, and so on. Given such plethora of information there is a pressing need to cross reference information across textual services. For example, commonly we read a news item and we wonder if there are any blogs reporting related content or vice versa.

In this paper, we present techniques to automate the process of cross referencing online information content. We introduce methodologies to extract phrases from a given "*query document*" to be used as queries to search interfaces with the goal to retrieve content related to the query document. In particular, we consider two techniques to extract and score key phrases. We also consider techniques to complement extracted phrases with information present in external sources such as Wikipedia and introduce an algorithm called RelevanceRank for this purpose.

We discuss both these techniques in detail and provide an experimental study utilizing a large number of human judges from Amazons's Mechanical Turk service. Detailed experiments demonstrate the effectiveness and efficiency of the proposed techniques for the task of automating retrieval of documents related to a query document.

1. INTRODUCTION

Content generated by individuals on the web is proliferating at a rapid pace including information produced on web pages, information on blogs and social networking sites or micro-blogging services. It is estimated that more than 60M blogs exist. Moreover, by the end of 2009 estimates report a total of 230M people on social networking sites [39]. The amount of information generated on a daily basis is unprecedented.

At the University of Toronto we have been building BlogScope (`www.blogscope.net`), a social media monitoring platform [2, 3, 4]. The system crawls, cleans, and aggregates data from several sources including blogs, top international news papers, major social networks, and collaborative wikis. An interactive search and analysis interface is provided to easily query and correlate this information. At the time of writing, the system is indexing over 425 million text documents increasing at the rate of 11 new documents per second, and serves visitors from over 20,000 unique IPs daily.

The proliferation of user generated content yields new opportunities to obtain information and to remain informed with developments in diverse domains such as politics, local and world news, technology etc. RSS readers have been very successful in delivering feeds of news or blogs. In light of this plurality of content there is a pressing need to cross reference information across sources. Consider for example, a news article from New York Times (either at the nytimes.com site or delivered through an RSS reader) reporting on a breaking news story or on current affairs. It is desirable to cross reference the information reported in the news source with the buzz in blogosphere and other user generated content sources. Commonly the reverse might be true, namely after reading a blog post, to check news sites to obtain additional information.

In this paper we present a solution to this problem which we refer to as *Query by Document* (QBD) allowing the user to submit a text document as a query and identify related documents from another text corpus. To provide such functionality, we present techniques to process *text* documents on demand and extract *key phrases* which are used to query BlogScope for retrieving blog posts related to the query document. Extracting key phrases from a text document to be used as queries is a challenging problem. We would like such phrases to convey the 'meaning' of the document but at the same time distinguishing enough to capture specific events or entities of interest unique to the document. Further, our approach is domain invariant (we use data from blogs and news sources just as a convenience).

Figure 1 shows an screenshot of the user interface for QBD. After the user inputs the text, the system shows a slider bar, which can be adjusted to match the required level of relevance for querying related documents. The slider bar can be set to "Very General" meaning fetch additional documents with somewhat similar content, or "Highly Specific" meaning fetch documents that talk about exactly the same events and topics. Clicking on one of "blogs", "news", or "web" retrieves back matching documents from the selected domain.

Further, we show that it is possible to extend these ideas towards the development of additional query types. In particular we utilize a large collection of pages from Wikipedia to extract phrases to enhance or substitute key phrases extracted from the input text document. This version of query by document, referred to as QBD-W, possesses highly novel semantics which we detail in Section 5.

WSDM'09, February 9–12, 2009, Barcelona, Spain.

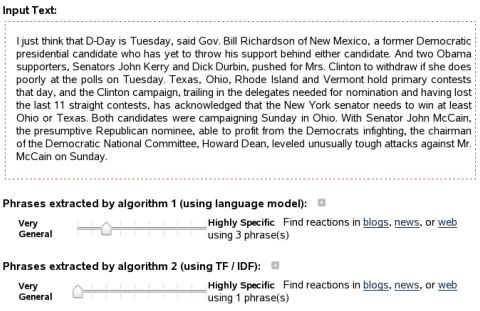


Figure 1: Screenshot for QBD user interface.

We present algorithm *RelevanceRank* to select candidate phrases from Wikipedia for this purpose. We validate the utility of our techniques by submitting extensive sets of results to Amazon's Mechanical Turk (MTurk)[27] for human evaluation. This enables users at large to act as judges of the quality of our findings. We present detailed experimental results validating the applicability of our approach.

To summarize, our main contributions are

- We define the problem of querying a text corpus using another text document. Such functionality can be used for correlating multiple informations sources.
- We formalize the problem of extracting relevant phrases from a document for the purpose of constructing queries to search engines (QBD). Two variants for solving this problem are proposed.
- We introduce the notion of using external knowledge sources (Wikipedia in our case) for the purpose of enhancing the set of query phrases. An algorithm for selecting relevant nodes from the Wikipedia graph is presented.
- We evaluate our techniques by employing human judges from Amazon's MTurk.

The rest of the paper is organized as follows. Section 2 reviews related work and provides necessary background. Section 3 formally defines the problems of interest. Section 4 details our phrase extraction and matching techniques. Algorithms for query enhancement are discussed in Section 5. Section 6 presents a comprehensive set of experiments validating the efficiency and effectiveness of the proposed techniques. Section 7 concludes the paper.

2. RELATED WORK

Relevance Feedback QBD is a method that enables retrieval of documents related, precisely or more generally to a query document. *Relevance feedback* (RF) [30], is a well-studied technique to improve query relevance which involves multiple interactions between an individual and the keyword search system to iteratively refine the search results. Specifically, in the initial step, one issues a query Q_0 which consists of a set of keywords, and the system returns its results R_0, comprised of a set of documents. Often, Q_0 may not precisely express search intentions, consequently, only some documents (forming a subset R_0^+ of R_0) are considered relevant, and others (denoted as $R_0^- \subset R_0$) are not. RF requires one to identify R_0^+ and R_0^-, and provide them as feedback. The iterative query refinement process continues until no more feedback is provided. Most solutions are based on Rocchio's formula [30], the basic idea of which is to modify the query by increasing (decreasing) the weights assigned to keywords which appear in relevant (irrelevant) documents, respectively; meanwhile, the query is expanded to include all terms present in relevant documents (note that negative term weights are ignored). Several variations and enhancements, most notably by Harman [17] and Ide [19], have been proposed for the Rocchio's approach.

RF has been utilized to display a "similar pages" link along with search results [32] in a web search scenario. While this functionality bears some resemblance with QBD, there are two fundamental differences between the two problems. First in RF there is an element of interaction, progressively refining the query after examination of search results. In general it has been demonstrated (e.g., [1]), that RF is more effective in enhancing *recall* than precision of the results. On the other hand QDB has the ability to identify both highly specific and generally related information utilizing keywords that are not necessarily present in the query document. Second, RF always has a query as a starting point, which progressively gets refined. In QDB the starting point in an entire document and there is no feedback process for refinement, i.e., the system has additional content at the beginning to utilize in order to specify the retrieval task.

In fact, a direct application of Rocchio's method to QBD (i.e., by setting Q_i and R_i^- to empty, and R_i^+ to contain only the query document) reduces to answering a keyword query comprised of all terms appearing in the query document, which will retrieve back only documents completely containing the original query text. Straightforward modifications of this method, such as taking only the top few terms with highest TF/IDF weights do not yield satisfactory results either: often, the scope of the results are so narrow that they provide little more than the information already present in the query document.

Phrase Extraction Another area related to our work is automatic extraction of phrases from documents. Existing solutions can be roughly classified into two categories. The first mainly aims at obtaining high result accuracy. These methods typically apply statistical machine learning techniques, involving expensive computation, and often rely on the availability of a large, high-quality training set. Specifically, several approaches [36, 34] employ a supervised or semi-supervised learning framework using training data to identify suitable key phrases. The main disadvantage is that they are not suitable for general texts as they require training. Recently such techniques have been enhanced with relationships between phrases which are then incorporated in the learning framework [23, 24]. This enhancement however requires additional training. Pantel et. al., [28] consider the term extraction problem using a corpus. They apply several tests for word association [22] utilizing prior information (from the corpus) regarding word co-occurrence counts. This is computationally intensive and requires several pruning thresholds which are difficult to set a priori in a document independent fashion. Tomokiyo et. al., [33] employ language models using a foreground and a background corpus, which after construction they combine in an additive way. This approach however, besides the requirement of multiple corpuses, is difficult to apply for phrases longer than two terms as it requires execution of an a priori style algorithm (to obtain word co-occurrence counts) which is prohibitively expensive.

The second category (e.g., [12], [26], [35]) are practical, efficient solutions proposed in the IR community. Typically, they employ fast, heuristic methods based on various statistics collected from the input document. Syntactic information has been utilized to identify candidate terms [9, 14]. In [9] candidate syntactic constructs were grouped together; each group consisting of sets of words with

the same head word. Then within each group term sequences were sorted by size. Maximal phrases within each group were subsequently identified. In [14] this idea is extended further, but this time the frequency of words in the document is considered. The technique identifies phrases, taking into account the syntactic label of words surrounding a term; ranking however does not utilize any sources or prior statistics.

Our phrase extraction techniques fall in the second category since we seek a practical and computationally tractable approach to implement in the BlogScope system. However, our focus is not to extract a set of phrases to summarize the document, but to construct a query. Moreover, we wish to provide flexibility regarding the relevance of the results. To give an example, consider a query document talking about the wedding of French president Nicolas Sarkozy with Carla Bruni. The ideal collection of phrases extracted for QBD would be the ordered set {"France", "Nicolas Sarkozy", "Carla Bruni"}. For this set, use of only the first phrase will bring back all documents talking about France (which is somewhat related to the query document), while use of all three phrases will search for more closely related documents. To our knowledge, none of existing methods for phrase ranking (e.g., [25]) are designed to suit this purpose. Moreover, such phrases are automatically extracted from the document at hand, providing the option to query for highly specific related content. Among a variety of choices, in our experiments we use the Yahoo Phrase Extractor [37] for the purpose of comparison. Besides providing easier repeatability of the experiments, the comparison with a leading commercial system underlines the value of the proposed solutions.

Query Enhancement We follow recent efforts to develop search and information discovery tools [38, 11, 8] that greatly benefit from Wikipedia's authoritative source of common knowledge by exploiting its graph structure, disambiguation pages, hierarchical classification, and concept co-occurrence information. QBD-W, which enhances the initial phrase set with background knowledge extracted from Wikipedia, bears similarities with traditional query expansion [10], whereas the query is automatically expanded. Recent proposals, e.g., [21], have attempted incorporating the information provided by Wikipedia to strengthen query expansion techniques, with considerable success. However, the goals of query expansion and QBD-W are very different. Specifically, the underlying assumption in query expansion is that one is unable to identify the precise keywords to express one's needs, and thus enhances the query with vague or surrounding concepts. The search engine, therefore, tries to decode the user's true intentions from this imprecise information, and locate the correct terms to enlarge the query. In contrast, QBD-W starts from phrases that best describe the query document, in the sense that they clearly express the one's knowledge; the goal is to locate new and related concepts to enrich that knowledge. In other words, query expansion aims to improve precision [10], while QBD-W also focuses on boosting recall. RelevanceRank, the proposed solution for QBD-W, is inspired by PageRank [18], TrustRank [16] and spreading activation framework [7]. A more in-depth comparison with these works is presented later in Section 5.

3. PROBLEM DEFINITION

Let $B = \{b_1, b_2, \ldots b_n\}$ be a set of n blogs. Since a blog is a reverse chronologically ordered stream of text posts written by an individual (or a group of individuals), we model a blog b as a sequence of P_b posts. Each post $p \in P_b$ contains two basic attributes: the textual document content $p.d$ and the timestamp signifying the creation time of the post $p.ts$.

For each blog b, in addition to P_b, BlogScope extracts and maintains information regarding the blog creation time, profile information regarding the blogger including age, profession, gender and geographical location (at the city, state, country level), as well as the aggregate count of in-links to the blog. The in-link count, along with several other properties computed on the posts are used subsequently by ranking algorithms. BlogScope adopts the usual text based search interface to issue keyword or phrase queries with conjunctive or disjunctive semantics. Moreover queries can be restricted temporally (e.g., only search blogs with posts between Jan 16th and 23rd 2007) with further restrictions on the blogger profile information (e.g., age between 30 and 35 and based in New Jersey, etc). All qualifying posts are searched for matches and those yielding a match are ranked using BlogScope's ranking mechanisms and subsequently returned as answers.

A QBD query q consists of a *query document* d, and optionally, temporal or other metadata restrictions (e.g., age, profession, geographical location) specified by the user. The specific challenge we address is the extraction of a number k (user specified) of phrases from d in order to form a query with conjunctive semantics. Ideally we would like them to be the phrases that an average user would extract from d to retrieve blog posts related to the document.

PROBLEM 3.1 (QBD). *Given a query document d, extract a user specified number k of phrases to be used as input query with conjunctive semantics to BlogScope. The blog posts retrieved should be rated by an average user as related to the content of the query document.*

All phrases extracted by QBD are present in the document. This functionality can be extended by taking into account external information sources. In particular Wikipedia contains a vast collection of information, in pages which exhibit high link connectivity. Consider the graph G_w extracted from Wikipedia in which each node v_i corresponds to the title of the i-th Wikipedia page and is adjacent to a set of nodes corresponding to the titles of all pages that the i-th page links to. We extracted such a graph, which we maintain up to date, currently consisting of 7M nodes. G_w encompasses rich amounts of information regarding phrases and the way they are related. For example starting with the node for 'Bill Clinton' we get links to nodes for the 'President of the United States', 'Governor of Arkansas', and 'Hillary Rodham Clinton'. This graph evidently provides the ability to enhance or substitute our collection of phrases extracted by QBD with phrases not present in the query document. Given the numerous outlinks from the 'Bill Clinton' page, it is natural to reason regarding the most suitable set of title phrases to choose from Wikipedia. Let v_i, v_l be two nodes in G_w corresponding to two phrases in the result of QBD for a document. Intuitively we would like phrases in G_w corresponding to nodes immediately adjacent to v_i and v_l to have higher chances to be selected as candidates for enhancing or substituting the result of QBD. This intuition is captured by an algorithm called RelevanceRank which we propose in Section 5.

The choice to enhance or substitute the results of QBD on a document with Wikipedia phrases depends on the semantics of the resulting query. For example consider a document describing an event associated with 'Bill Clinton', 'Al Gore' and the 'Kyoto Protocol' and that these three phrases are the result of QBD on a document. If we add the phrase 'Global Warming' extracted from Wikipedia (assuming that this phrase in not present in the result of QBD) we will be retrieving blog posts possibly associating 'Global Warming' with the event described in the query document (if any)[1].

[1]Incidently all three Wikipedia pages for Bill Clinton, Al Gore and the Kyoto Protocol point to the Wikipedia page for Global Warming.

As an additional example consider a document concerning a new movie released by Pixar animation studios (say Ratatouille); assume that this document does not mention any other animated movies produced by Pixar. Nodes corresponding to other animated movies produced by 'Pixar' would be good candidates from Wikipedia since they are pointed by both the node for 'Pixar' and the node for 'Ratatouille'. By substituting (all or some) of the phrases in QBD by phrases extracted from Wikipedia, such as 'Toy Story' and 'Finding Nemo', we would be able to retrieve posts related to other movies produced by 'Pixar'. All the above intuitions are formalized in the following problem:

PROBLEM 3.2 (QBD-W). *Given a set of phrases C_{qbd} extracted by QBD containing k phrases from d, identify a user defined number k' of phrases utilizing the result of QBD and the Wikipedia graph G_w. The resulting k' phrases will be used as input query with conjunctive semantics to BlogScope. The blog posts retrieved should be rated by an average user as related to the content of the query document.*

4. PHRASE EXTRACTION: QBD

In this section we detail the methodology for solving the QBD problem introduced earlier. The basic workflow behind our solutions to QBD is as follows: (1) Identify the set of all candidate key phrases C_{all} for the query document d. (2) Assess the significance of each candidate phrase $c \in C_{all}$, assigning a score $s(c)$ between 0 and 1. (3) Select the top-k (for a user specified value of k) phrases as C_{qbd} as a solution to QBD. We detail generation of candidate key phrases C_{qbd} in Section 4.1 and the mechanisms to score candidate key phrases in Section 4.2. Our solution to QBD-W and the algorithm *RelevanceRank* are presented in Section 5.

4.1 Extracting Candidate Phrases

We extract candidate phrases C_{all} from the query document d with the help of a *part-of-speech tagger* (POST) [29]. Specifically, for each term $w \in d$, POST determines its part-of-speech (e.g., noun, verb, adjective, etc.) by applying a pre-trained classifier on w and its surrounding terms in d. For instance, in sentence "Wii is the most popular gaming console", term "Wii" is classified as a noun, "popular" as an adjective, and so on. We represent the tagged sentence as "Wii/N is/V the/P most/A popular/J gaming/N console/N", where "N", "V", "P", "A", and "J" signify noun, verb, article, adverb, and adjective respectively.

Based on the part-of-speech tags, we consider all *noun phrases* as candidate phrases, and compute C_{all} by extracting all such phrases from d. A noun phrase is a sequence of terms in d whose part-of-speech tags match a *noun phrase pattern* (*NPP*). Table 1 lists some example NPPs used and their instances (i.e., noun phrases). To facilitate efficient identification of candidate phrases, the pattern set PS is organized as a forest of tries as illustrated in Figure 2. Each path (e.g., $n_{10} - n_{11} - n_{14}$) ending with symbol '$' signifies a pattern in PS (e.g., 'NN').

EXAMPLE 4.1. *Let d = "Wii is the most popular gaming console". First we obtain the POST tags for each word in d, obtaining the tagged document "Wii/N is/V the/P most/A popular/J gaming/N console/N". The corresponding tag sequence d_{POS} is thus "NVPAJNN". We then scan d_{POS}, and match all subsequences of length at most 5 against the PS trie forest (see Figure 2). Continuing the example, the first tag "N" matches node n_{10}; since n_{10} has '$' ($n_{13}$, signifying end-of-pattern) as a child, the corresponding term c_1 = "Wii" in d is identified as a candidate phrase. Similarly, the subsequence "JNN" and "NN" match the paths $n_1 - n_2 - n_5$ and $n_{10} - n_{11}$ respectively, thus c_2 = "popular gaming console" and c_3 = "gaming console" are extracted as candidate phrases. Note that we do not require a candidate phrase to be* maximal*, for instance, c_3 is a subsequence of c_2. This redundancy is eliminated later through scoring and top-k selection of candidate phrases, detailed next.*

Pattern	Instance
N	Nintendo
JN	global warming
NN	Apple computer
JJN	declarative approximate selection
NNN	computer science department
JCJN	efficient and effective algorithm
JNNN	Junior United States Senator
NNNN	Microsoft Host Integration Server
...	...
NNNNN	United States President George Bush

Table 1: Example of Noun Phrase Patterns and Instances

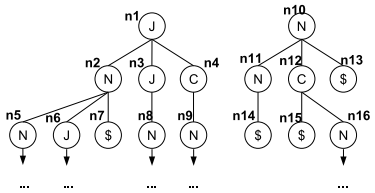


Figure 2: PS Trie Forest.

4.2 Scoring Candidate Phrases

Once all candidate phrases are identified as C_{all}, a scoring function f is applied to each phrase $c \in C_{all}$. The scoring function assigns a score to c based on the properties of c, taking into account both the input document, and the background statistics about terms in c from the BlogScope corpus. The candidate phrases are revised in a pruning step to ensure that no redundant phrases are present. We propose two scoring mechanisms, f_t and f_l for this purpose. f_t utilizes the TF/IDF information of terms in c to assign a score, while f_l computes the score based on the *mutual information* of the terms in phrase c. Both ranking mechanisms share the same pruning module to eliminate redundancy in the final result C_{qbd}. We provide examples and experimental comparison of these two approaches later in this paper.

4.2.1 TF/IDF Based Scoring

We first describe f_t, which is a linear combination of the total TF/IDF score of all terms in c and the degree of *coherence* of c. Coherence quantifies the likelihood these terms have in forming a single concept. Formally, let $|c|$ be the number of terms in c; we use $w_1, w_2, \ldots, w_{|c|}$ to denote the actual terms. Let $idf(w_i)$ be the inverse document frequency of w_i as computed over all posts in BlogScope's corpus. f_t is defined as

$$f_t(c) = \sum_{i=1}^{|c|} tfidf(w_i) + \alpha \cdot coherence(c) \qquad (4.1)$$

where α is a tunable parameter.

The first term in f_t aggregates the importance of each term in c. A rare term that occurs frequently in d is more important than a common term frequently appearing in d (with low idf, e.g., here, when, or hello). This importance is nicely captured by $tfidf$ for the term [31][2]. We use the total, rather than average $tfidf$ to favor phrases that are relatively long, and usually more descriptive.

[2]For implementation purposes we use $tfidf(w) = tf(w) \cdot idf(w)^2$

The second term in f_t captures how coherent the phrase c is. Let $tf(c)$ be the number of times c appears in the document d, the coherence of c is defined as

$$coherence(c) = \frac{tf(c) \times (1 + \log tf(c))}{\frac{1}{|c|} \times \sum_{i=1}^{|c|} tf(w_i)} \quad (4.2)$$

Intuitively, Equation 4.2 compares the frequency of c (the numerator) against the average TF of its terms (the denominator). The additional logarithmic term strengthens the numerator, preferring phrases appearing frequently in the input document. For example, consider the text fragment "... at this moment Dow Jones ...". Since the phrase "moment Dow Jones" matches the pattern "NNN", it is included in C_{all}. However it is just a coincidence that the three nouns appear adjacent, and "moment Dow Jones" is not a commonly occurring phrase as such. The coherence of this phrase is therefore low (compared to the phrase "Dow Jones"), since the tf of the phrase is divided with the average tf of terms constituting it. This prohibits "moment Dow Jones" to appear high in the overall f_t ranking.

Based on TF/IDF scoring, f_t is good at distinguishing phrases that are characteristic of the input document. In the running example d = "Wii is the most popular gaming console", f_t strongly favors "Wii" over "gaming console" since the former is a much rarer term and thus has a much higher idf score. However, f_t also has the drawback that it is often biased towards rare phrases.

4.2.2 Mutual Information Based Scoring

f_l uses mutual information (MI) between the terms of c as a measure of coherence in the phrase c along with idf values from the background corpus. Mutual information is widely used in information theory to measure the dependence of random variables. Specifically, the pointwise mutual information of a pair of outcomes x and y belonging to discrete random variables X and Y is defined as [5]

$$PMI(x, y) = \log\left(\frac{prob(x, y)}{prob(x)prob(y)}\right) \quad (4.3)$$

where $prob(x), prob(y), prob(x, y)$ are the probability of x, y and the combination of the two respectively. Intuitively, for a phrase c consisting of terms $w_1, w_2, ..., w_{|c|}$, the higher the mutual information among the terms, the higher are the chances of the terms appearing frequently together; and thus they are more likely to be combined to form a phrase.

The scoring function f_l takes a linear combination of idf values of terms in c, frequency of c, and the pointwise mutual information among them. Let $tf(c)$ and $tf(POS_c)$ be the number of times c and its part-of-speech tag sequence POS_c appear in d and POS_d respectively, then

$$f'_l(c) = \sum_{i=1}^{|c|} idf(w_i) + \log\frac{tf(c)}{tf(POS_c)} + PMI(c) \quad (4.4)$$

The first part in the equation above represents how rare or descriptive each of the terms in c is. The second part denotes how frequent the phrase c is at the corresponding POS tag sequence in the document. The third part captures how likely are the terms to appear together in a phrase.

The $PMI(c)$ for a phrase c is

$$PMI(c) = \log\left(\frac{prob(c)}{\Pi_{i=1}^{|c|} prob(w_i)}\right)$$

PMI can be evaluated either at the query document itself or at the background corpus. Computation of these probabilities for the background corpus requires a scan of all documents, which is prohibitively expensive[3]. In order to compute PMI using d only, let $prob(w_i)$ and $prob(c)$ denote the probability of occurrence of w_i and c respectively at the appropriate part-of-speech tag sequence.

$$prob(c) = \frac{tf(c)}{tf(POS_c)}, \text{ and } prob(w_i) = \frac{tf(w_i)}{tf(POS_{w_i})}$$

Substituting these probabilities in Equation 4.4,

$$f'_l(c) = \sum_{i=1}^{|c|} idf(w_i) + \log\frac{tf(c)}{tf(POS_c)} + \log\left(\frac{\frac{tf(c)}{tf(POS_c)}}{\Pi_{i=1}^{|c|}\frac{tf(w_i)}{tf(POS_{w_i})}}\right) \quad (4.5)$$

The scoring function as defined in Equation 4.5 identifies how rare or descriptive each term is and how likely these terms are to form a phrase together. This definition however does not stress adequately the importance of how frequent the phrase is in document d; therefore we weight it by $\frac{tf(c)}{tf(POS_c)}$ before computing the final score f_l. The scoring function f_l therefore is,

$$f_l(c) = \frac{tf(c)}{tf(POS_c)} \times \left(\sum_{i=1}^{|c|} idf(w_i) + \log\frac{tf(c)}{tf(POS_c)} + \log\left(\frac{\frac{tf(c)}{tf(POS_c)}}{\Pi_{i=1}^{|c|}\frac{tf(w_i)}{tf(POS_{w_i})}}\right)\right) \quad (4.6)$$

The tf values in the above equations are computed by scanning the document d once, while the idf values are maintained precomputed for the corpus.

The scoring function (f_t or f_l) evaluates each phrase $c \in C_{all}$ individually. As a result, candidate phrases may contain redundancy. For example, a ranking function may judge that both c_1 = "gaming console" and c_2 = "popular gaming console" as candidate phrases. Since c_1 and c_2 refer to the same entity, intuitively only one should appear in the final list C_{qbd}. We therefore apply a post-processing step after evaluating the ranking function on elements of C_{all}. Methodology for computing C_{qbd} is shown in Algorithm 1. Lines 7-14 demonstrate the pruning routine after evaluating the ranking function. Specifically, a phrase c is pruned when there exists another phrase $c' \in C_{qbd}$ such that (i) c' has a higher score than c, and (ii) c' is considered redundant in presence of c. The function *Redundant* evaluates whether one of the two phrases c_1, c_2 is unnecessary by comparing them literally.

Note that sometimes the shorter phrase may be more relevant, so we should not simply identify longer phrases. For instance, the phrase "drug" may have higher score than a longer phrase "tuberculosis drugs" in a document that talks about drugs in general, and tuberculosis drugs is one of the many different phrases where the term "drug" appears. Also, the candidate set C_{all} may contain phrases with common suffix or prefix, e.g., "drug resistance", "drug facility" and "drug needs", in which case we keep only the top few highest scoring phrases to eliminate redundancy. *Redundant* returns *true* if and only if either one phrase subsumes the other, or multiple elements in C_{qbd} share common prefix/suffix.

which is widely used in practice, most notably by the Apache Lucene's Similarity class.

[3]If statistics regarding the co-occurrence in the text of n ($n \geq 2$) terms is available one can obtain more precise correlation information for sets of terms. However this comes with a high computation, storage and maintenance cost as the number of all such combinations of keywords is very high. In BlogScope we maintain precomputed idf values for single keywords only, and statistics for combinations of keywords is not materialized.

Algorithm 1 Algorithm for QBD

INPUT document d, and required number of phrases k

ComputeQBD

Run a POS tagger to obtain the tag sequence POS_d for d
Initialize C_{all} and C_{qbd} to empty
Match POS_d against the PS Trie forest
For each subsequent $POS_c \subset POS_d$ that matches a NPP, append the corresponding term sequence to C_{all}
for each $c \in C_{all}$ **do**
  Compute the score s_c using either of f_t or f_l
  if NOT exists $c' \in C_{qbd}$ such that ($Redundant(c, c') = true$ and $s_{c'} > s_c$) **then**
    Add c to C_{qbd}
  end if
  for each $c' \in C_{qbd}$ **do**
    if $Redundant(c, c')$ and $s_{c'} < s_c$ **then**
      Remove c' from C_{qbd}
    end if
  end for
  If $|C_{qbd}| > k$, remove the entry with minimum score
end for
OUTPUT C_{qbd}

5. USING WIKIPEDIA: QBD-W

We have constructed a directed graph $G_w =< V, E >$ by preprocessing a snapshot of Wikipedia, modeling all pages with the vertex set V and the hyperlinks between them with the edge set E. Specifically, a phrase c is extracted for each page p_c in Wikipedia as the title of the page. Each such phrase is associated with a vertex in V. Hyperlinks between pages in Wikipedia translate to edges in the graph G_w. For example, the description page for "Wii" starts with the following sentence: "The Wii is the fifth home video game console released by Nintendo", which contains hyperlinks (underlined) to the description pages of "video game console" and "Nintendo" respectively. Intuitively, when the Wikipedia page p_c links to another page $p_{c'}$, the underlying phrases c and c' are related. Consider two pages p_{c_1} and p_{c_2} both linking to $p_{c'}$. If the number of links from p_{c_1} to $p_{c'}$ is larger than the number of links from p_{c_2} to $p_{c'}$, we expect c_1 to have a stronger relationship with c'. This can be easily validated by observing the Wikipedia data.

Formally, the Wikipedia graph G_w is constructed as follows: a vertex v_c is created for each phrase c which is the title of the page p_c. A directed edge $e =< v_c, v_{c'} >$ is generated if there exists a hyperlink in p_c pointing to $p_{c'}$. A numerical weight wt_e is assigned to the edge $e =< v_c, v_{c'} >$ with value equal to the number of hyperlinks from p_c pointing to $p_{c'}$. We refer to the weight of the edge between two vertices in graph G_w as their *affinity*.

EXAMPLE 5.1. *Figure 3 depicts the interconnection between phrases c_1 = "Wii", c_2 = "Nintendo", c_3 = "Sony", c_4 = "Play Station", and c_5="Tomb Raider", in the Wikipedia graph. The number beside each edge signifies its weight, e.g., $wt_{<c_1,c_2>} = 7$ implying that there are 7 links from the description page of "Wii" to that of "Nintendo". Node c_2 is connected to both c_1 and c_3, signifying that "Nintendo" has affinity with both "Wii" and "Sony". Edge $< c_2, c_1 >$ has a much higher weight than $< c_2, c_3 >$, signifying that the affinity between "Nintendo" and "Wii" is stronger than that between "Nintendo" and "Sony" (the manufacturer of Play Station 3, a competitor of Wii). Therefore, if "Nintendo" is an important phrase mentioned in the input document d, i.e., $c_2 \in C_{qbd}$, it is much more likely that c_1 (rather than c_3) is closely relevant to d, and thus should be included in the enhanced phrase set after QBD-W.*

Once G_w is ready and the set C_{qbd} is identified, it can be enhanced using the Wikipedia graph according to the following procedure: (1) Use C_{qbd} to identify a seed set of phrases in the Wikipedia graph G_w. (2) Assign an initial score to all nodes in G_w. (3) Run the algorithm *RelevanceRank* as described in Algorithm 2 to iteratively assign a relevance score to each node in G_w. The *RelevanceRank* algorithm is an iterative procedure in the same spirit as biased PageRank [18] and TrustRank [16]. (4) Select the top-k' highest scoring nodes from G_w (for user specified value of k') as top phrases C_{wiki}.

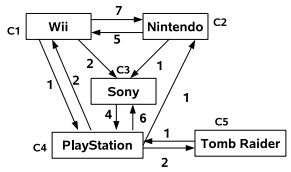


Figure 3: Part of Wikipedia graph with five nodes

Algorithm 2 Algorithm to compute RelevanceRank

INPUT Graph $G_w =< V, E >$, QBD phrases C_{qbd}, k'

RelevanceRank

Initialize the seed set to empty set
for each $c \in C_{qbd}$ **do**
  Compute node $v \in V$ with smallest edit distance to c
  If $edit\_distance(c, v) < \theta$, add v to S
end for
for each $v \in V$ **do**
  Assign initial score to v based on Equation 5.1
end for
for $i = 1$ **to** $MaxIterations$ **do**
  Update scores for each $v \in V$ using Equation 5.3
  If convergence, i.e., $RR^i = RR^{i-1}$, break the for loop
end for
Construct C_{wiki} as the set of top-k' vertices with highest RR scores

This initializes the scores of all vertices not in the seed set to zero. Scores of vertices in the seed set the normalized to lie in $[0, 1]$ such that the sum is 1.

Next *RelevanceRank* iterates (Lines 9-12) until convergence or reaching a maximum number of iterations *MaxIterations*. The i^{th} iteration computes RR^i based on the results of RR^{i-1} following the *spreading activation* framework [7]. Specifically, the *transition matrix* T is defined as

$$T[v, v'] = \begin{cases} \frac{wt_e}{\sum_{e'=(v,w)} wt_{e'}} & \text{if } \exists e =< v, v' > \in E \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

The entry $T[v, v']$ represents the fraction of out-links from the page corresponding to v in Wikipedia that point to the page associated with v'. Observe that each entry in T is in range $[0, 1]$ and the sum

	Wii	Sony	Nintendo	Play Station	Tomb Raider
Wii	0	2/10	7/10	1/10	0
Sony	0	0	0	4/4	0
Nintendo	5/6	1/6	0	0	0
Play Station	2/11	6/11	1/11	0	2/11
Tomb Raider	0	0	0	1/1	0

Table 2: Transition matrix for graph in Figure 3

of all entries in a row is 1. Conceptually T captures the way a vertex v passes its affinity to its neighbors, so that when v is relevant, it is likely that a neighboring phrase v' with high affinity to v is also relevant, though to a lesser degree.

EXAMPLE 5.2. *The transition matrix for vertices in Figure 3 is displayed in Table 2.*

To model the fact that a phrase connected to nodes from C_{qbd} through many intermediate nodes is only remotely related, the propagation of RR is dampened as follows: with probability α_v, v passes its RR score to its successors, and with probability $(1-\alpha_v)$ to one of the seed vertices S. Formally RR_v^i in the ith iteration is computed by

$$RR_v^i = \sum_{e=<v',v>} \alpha_{v'} \cdot RR_{v'}^{i-1} \cdot T[v',v] + RR_v^0 \sum_{v' \in V} (1-\alpha_{v'}) RR_{v'}^{i-1} \quad (5.3)$$

The first term in the equation represents propagation of RR scores via incoming links to v. The second term accounts for transfer of RR scores to seed nodes with probability $1-\alpha_{v'}$. Recall that RR_v^0 is zero for phrases not in the seed set, and thus the second term in the equation above is zero for $v \notin S$.

The $RelevanceRank$ algorithm can be alternatively explained in terms of the random surfer model. In the Wikipedia graph G_w, first the seed nodes are identified by using the result C_{qbd} of QBD. Each of these seed nodes is assigned an initial score using a scoring function (f_t or f_l). All other nodes are assigned score zero. The surfer starts from one of the seed nodes. When at node v, the surfer decides to continue forward, selecting a neighboring node v' with probability $\alpha_v \cdot T[v,v']$. With probability $1-\alpha_v$, the surfer picks a node at random from the initial seed set. The probability of selection of the node from the seed set is proportional to the initial RR^0 scores of the nodes in S. At convergence, RR score of a node is the same as the probability of finding the random surfer there.

In RelevanceRank, with probability $1-\alpha_v$, the random surfer jumps back to nodes in the seed set only and not to any node in G_w. This is in similar spirit as the topic-sensitive PageRank and TrustRank algorithms [18, 16], which use a global constant value $\alpha_v = \alpha$ for all $v \in G_w$ for returning back to one of the seed nodes. Selection of a constant α is however not suitable for RelevanceRank for the following two reasons:

- The RelevanceRank scoring function must prefer nodes that are close to the initial seed set. In TrustRank, existence of a path between two nodes suffices for propagation of trust (as stationary state probabilities are probability values after the surfer makes infinitely many jumps). The same holds true for PageRank as well, where existence of a path is sufficient for propagation of authority. For the case of RelevanceRank however, the length of the path is an important consideration. Propagation of RR scores over long paths needs to be penalized. Only nodes in the vicinity of seed nodes are relevant to the query document. The value of α_v therefore must depend on the distance of a node from the seed set.

- G_w consists of over 7 million nodes. Execution of the iterative algorithm to compute RR scores over the entire graph for every query is not feasible. Unlike TrustRank or PageRank, where one-time offline computation is sufficient, RelevanceRank needs to be evaluated on a per-query basis. Since only nodes close to the seed set are relevant, we set α_v to zero for vertices $v \in V$ far from the seed set S. Let l_{max} be the maximum permissible length of path from a node to S. Define the graph distance $GD(v)$ of a node v as its distance from the closest node in the seed set. Formally,

  $$GD(v) = min_{v' \in S} distance(v', v)$$

  where $distance$ represents the length of the shortest path between two nodes. Thus, if $GD(v) \geq l_{max}$ for some $v \in V$, α_v is assigned value 0. Application of this restriction on α_v allows us to chop off all nodes from G_w that are at distance greater than l_{max} from S, which significantly reduces the size of the graph we need to run the $RelevanceRank$ algorithm on. As the value of l_{max} increases, the size of sub-graph over which RelevanceRank is to be computed increases, leading to higher running times.

We implemented the version of RelevanceRank with a constant value of α for the purpose of experimentation. Apart from high computational overhead, we discovered that this implementation always returned irrelevant set of phrases belonging to a densely connected clique far away from starting seed set. For example, query starting from any document related to George Bush returned protein names as result of QBD-W, since some research on proteins was conducted during Mr Bush's presidency and proteins form a highly dense subgraph in the Wikipedia graph. This is expected, since the stationary probabilities in the random surfer model used for PageRank is very high for nodes in such cliques independent of the starting node. Hence, we experimentally verified that the use of either of TrustRank or topic sensitive PageRank for this problem is not suitable.

For the above mentioned reasons, α_v for a node v is defined as a function of its graph distance $GD(v)$. We would like α_v to decrease as $GD(v)$ increases such that $\alpha_v = 0$ if $GD(v) \geq l_{max}$. We define α_v as

$$\alpha_v = max\left(0, \alpha_{max} - \frac{GD(v)}{l_{max}}\right) \quad (5.4)$$

for some constant $\alpha_{max} \in [0,1]$.

When the iterative algorithm for computation of RelevanceRank finishes, each node is assigned an RR score. The process is guaranteed to converge to a unique solution, as the algorithm is essentially the same as that of computing stationary state probabilities for an irreducible Markov chain with positive-recurrent states only [13]. These nodes, and thus corresponding phrases, are sorted according to the RR scores, and top-k' (for a user-defined value of k') are selected as the enhanced phrase set C_{wiki}. The new set C_{wiki} may contain additional phrases that are not present in C_{qbd}. Also, phrases from C_{qbd} included in C_{wiki} may have been re-ranked, that is the order of phrases in C_{qdb} appearing in C_{wiki} may be different than the corresponding order these phrases have in C_{qdb}. This means, even for $k' \leq k$, the set C_{wiki} can be very different from C_{qbd} depending on the information present in Wikipedia.

EXAMPLE 5.3. *Consider the graph in Figure 3. Assume that the seed set consists of only one node "Nintendo". Let $\alpha_{max} = 0.8$ and $l_{max=2}$. Then, initial score for Nintendo will be 1, $RR^0_{Nintendo} = 1$; and for Sony, Wii and Play Station, the initial score will be zero. Also, $\alpha_{Nintendo} = 0.8$, $\alpha_{Sony} = 0.3$, $\alpha_{Wii} = 0.3$, $\alpha_{PlayStation} = 0$, and $\alpha_{TombRaider} = 0$. Note that, the random surfer can never reach the node "Tomb Raider" in this setting since the surfer must jump back to "Nintendo" when he reaches*

iterations	Wii	Sony	Nintendo	Play Station
0	0	0	1	0
1	0.67	0.13	0.20	0
2	0.13	0.06	0.74	0.06
3	0.49	0.11	0.38	0.02
4	0.25	0.08	0.62	0.05
5	0.41	0.10	0.46	0.03
..	..	..	..	..
infinite	0.35	0.09	0.52	0.03

Table 3: RelevanceRank scores after 1-5 iterations and at convergence

the node "Play Station". Hence we can simply remove all nodes, including "Tomb Raider", with graph distance greater than 2 for calculating RR scores. The transition matrix is presented in Table 2. Only the first four rows and columns of the transition matrix are relevant. RelevanceRank scores after few iterations will be as displayed in Table 3. At convergence, "Nintendo" has the highest RR score 0.52, with "Wii" at the second position. Scores for "Sony" and "Play Station" are low as expected.

EXAMPLE 5.4. *Consider the news article titled "U.S. Health Insurers Aim to Shape Reform Process" taken from Reuters*[4]*. Top 5 phrases in QBD for this article consists of "america's health care system", "ahip's ignani", "special interests", "tax credits" and "poorer americans". While these phrases do relate to the meaning of the document, they do not necessarily constitute the best fit for describing it. The result of running QBD-W with the same value of $k' = k = 5$ results in "american health care", "ahip", "universal health care", "united states" and "poore brothers". Arguably, the latter articulates the theme of the document in a much better way. Enhancement using wikipedia graph has replaced and re-ranked most items from the seed set consisting of 5 initial terms. For example, the phrase "AHIP's Ignani" that appears thrice in the document, and which refers to the CEO Karan Ignani of America's Health Insurance Plans, has been replaced with just AHIP. Also, "america's health care system" is re-written as "american health care" (due to use approximate string matching) which is the title of a page in Wikipedia.*

6. EXPERIMENTS

We now present the experimental evaluation of our techniques. Section 6.1 discusses the data sets utilized for our experiments. Section 6.2 describes our use of Amazon Mechanical Turk to evaluate the quality of the phrase extraction techniques. Then, Section 6.3 presents our experiments for measuring the retrieval quality under our query-by-document scenario, demonstrating that our techniques significantly outperform existing, strong baselines.

6.1 Data Sets and Alternative Techniques

Query Documents: We extracted and present results for a random sample of 34 news articles from the New York Times, The Economist, Reuters, and Financial Times published during Aug - Sept, 2007. We refer to this set of documents as *NYTS*. These are the documents that we use as queries to locate related blog postings on BlogScope. We evaluated several collections of data sets and we present NYTS as representative of our results.

Techniques: We present two algorithms, QDB employing f_t scoring (referred to *QBD-TFIDF* in our experiments) and QBD employing f_l scoring (referred to *QBD-MI*) to extract query phrases from a document. These techniques extract a ranked list of the top-k phrases from a document. In our experiments, we present results for varying values of k. Similarly, we experimented with the extraction technique from Section 5, which identifies important terms that do not appear in the document, by analyzing the graph of Wikipedia; we refer to this technique as *QBD-W*. As a strong baseline, we use the "Yahoo Term Extraction" (*QBD-YAHOO*) service [37], which takes as input a text document and returns a list of significant words or phrases extracted from the document.

[4]http://www.reuters.com/article/domesticNews/idUSN2024291720070720

6.2 Quality of Phrase Extraction

Our first step is to examine the quality of the phrases obtained by the various techniques. For this, we run two experiments, using human annotators utilizing Amazon's Mechanical Turk service.

Annotator-Nominated Relevant Phrases: We wish to examine whether phrases identified by human annotators as important, are also identified as important from our techniques. To avoid any bias, we launched a large scale user study using the Amazon Mechanical Turk service [27]. This service offers access to a community of human subjects and tools to distribute small tasks that require human intelligence. In our study, each Mechanical Turk annotator was presented with a document, and had to list up to 10 phrases that are characteristic of the document; in particular we asked the annotators to identify phrases that they would "*use as queries to locate this document on Google*." To ensure the quality of this distributed labeling effort, we asked five distinct annotators to extract phrases from each document. Then, for each phrase identified, we measure how many users agreed that a given phrase is relevant, to compute the level of agreement across annotators for each phrase. Since our labelers might use slightly different wording to refer to the same phrase (e.g., phrases "crandon swat team" and "crandon swat" refer to the same entry), we use a string similarity technique [6] in conjunction with our own manual inspection, to group similar phrases together. Annotators did not have access to our techniques and had no knowledge of the output of our algorithms on the same documents for this experiment to avoid any bias.

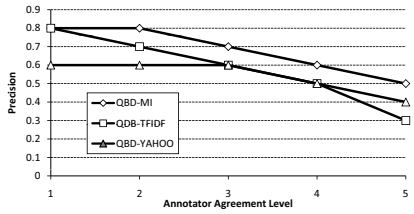


Figure 4: Precision against the annotator-nominated relevant phrases.

We compute the precision of each technique, against the pool of annotations produced by Mechanical Turk annotators. We define the precision at agreement level l for a technique $\mathcal{T}$, as $precision = h/k$, where h is the number of phrases identified by at least l humans *and* by technique $\mathcal{T}$ (i.e., the number of common phrases between humans and $\mathcal{T}$), and k is the total number of phrases extracted by technique $\mathcal{T}$. Figure 4 presents the precision of our techniques, for $k = 2$ for different levels of agreement. (The results are similar, qualitatively, for other values of k.) We observe that *QBD-MI* outperforms *QBD-TFIDF* and *QBD-YAHOO* across all agreement levels. This indicates that there exists good agreement between the output of our techniques and the expectations of humans.

Automatically Extracted Relevant Phrases: The previous experiment indicated that the phrases extracted by our techniques agree with the phrases that are independently extracted by humans. Our techniques, however, extracts a larger number of phrases that were not listed by the annotators. In this experiment, we expose the phrases to the annotators asking them the question: *are these phrases relevant or non relevant to the contents of the document*. When

the annotators look at the automatically extracted phrases, they can easily determine whether a particular phrase accurately depicts the contents of the document. So, to examine the accuracy of our techniques, we used each of *QBD-MI*, *QBD-TFIDF*, and *QBD-YAHOO*, and extracted from each document the set of top-k phrases ($k = 20$) for each technique. This resulted in a pool of a maximum of 60 phrases per document (usually the number was lower, as often the three techniques extracted similar sets of phrases). Each phrase in each pool was labeled by five annotators. At the end of the process, we associated each phrase with an agreement level, counting how many annotators marked the phrase as relevant.

Figure 5 presents the *precision@k* of our techniques, for varying values of k and for level of agreement 3 and above. We observed that *QBD-MI* outperforms the other techniques for $k < 4$, while the techniques tend to perform equally for larger values of k. Restricting our attention to a few phrases only, brings out differences among the techniques considered. For a large number of phrases the techniques appear similar in terms of characterizing the content of the document. Our primary goal however is to utilize such phrases to identify documents (in our case blog posts) related to the query document. We report on such experiments next.

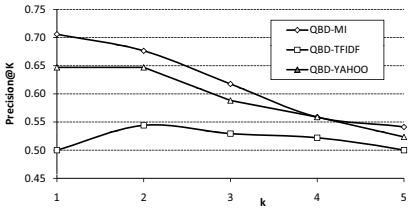
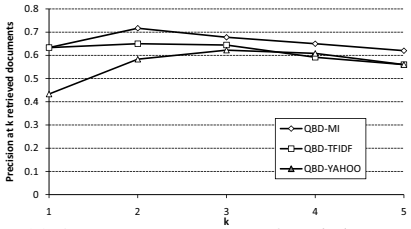
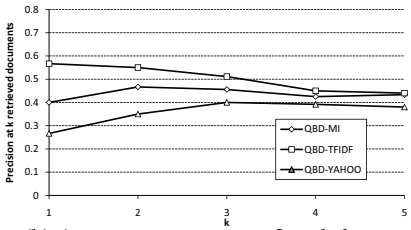


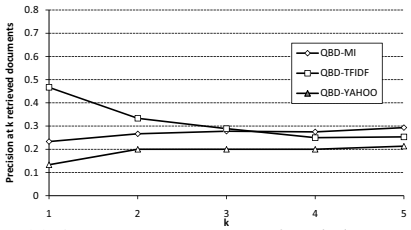
Figure 5: Precision after extracting the top-k phrases.



(a) Annotator agreement: 2 and above



(b) Annotator agreement: 3 and above



(c) Annotator agreement: 4 and above

Figure 6: Retrieval precision using k phrases ($precision@k$).

6.3 Quality of Document Retrieval

Quality of Querying-by-Document: We deploy our techniques to generate queries from the query document to be used as query phrases in order to retrieve related documents from BlogScope. To measure the quality of the various query-by-document techniques, for each "query-document" in *NYTS*, we extracted top-k phrases (for varying values of k) and then submitted these phrases as queries to the BlogScope search engine. We then retrieved the top-5 matching documents for these queries, and generated a pool of retrieved documents for each query-document and each value of k. The query-document and the pool of retrieved documents were submitted to Amazon Mechanical Turk, where we asked 5 annotators to examine individually each retrieved document and decide whether it is relevant to the query-document.

Figure 6 illustrates the results. One clear trend is that *QBD-MI* and *QBD-TFIDF* systematically outperform *QBD-YAHOO* for different annotator agreement levels and for different values of k (the number of phrases used as seed queries to BlogScope). This indicates that *QBD-MI* and *QBD-TFIDF* are able to identify documents (blog posts) more relevant to the query document. To understand why *QBD-YAHOO* performs worse than the other techniques, we manually inspected the queries and the corresponding retrieved documents. The *QBD-YAHOO* method tends to generate queries (phrases) that capture the general topical area of the document, but are not specific enough to retrieve documents that will be deemed relevant by the users. In fact, the extracted terms could be used as general-purpose tags for summarizing and classifying the document in a broad classification taxonomy but are relatively inadequate when used as queries due to their generality. Comparing *QBD-MI* and *QBD-TFIDF*, we can see that *QBD-MI* performs better for lower levels of annotator agreement, but this trend reverses when we consider only documents for which 3 and more annotators agree. This result indicates that *QBD-MI* is better in retrieval environments where users are looking for a diversity of results in the returned matches, while *QBD-TFIDF* is better suited for environments where the goal is to present results that are commonly accepted as being related to the topic of the query, ignoring potentially less common interpretations of the query-document topics.

We observed that extracting more than $k = 5$ phrases from a query document to utilize as queries to BlogScope was never required for the case of NYTS set. In fact it was not possible to retrieve related blog posts when queries consisted of more than five phrases (the returned result was empty; evidently no article in our NYTS collection appeared verbatim in the Blogosphere).

Quality of Querying-by-Document-Wikipedia: Finally, we run a set of experiments using *QBD-W*, that utilizes the Wikipedia expansion technique, described in Section 5, to identify useful query terms for our query-by-document approach. We conducted an experiment using Mechanical Turk similar to that for assessing the quality of QBD asking annotators to characterize the resulting documents (blog posts) as relevant or not to the query document. Figure 7 reports obtained precision for different agreement levels as the number of phrases k' used for retrieving the documents is varied. The type of query results produced by this technique tend to be distinct and different than the results returned by *QBD-MI*, *QBD-TFIDF*, and *QBD-YAHOO*. In particular, the fact that *QBD-W* generates query terms that do not appear in the document makes the returned results more "serendipitous"; the returned documents capture a more implicit notion of relevance, and users tend to prefer

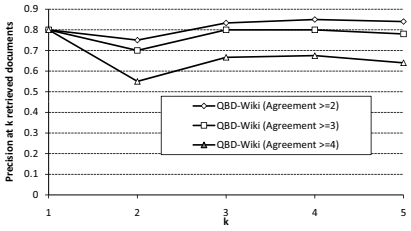


Figure 7: Retrieval precision of *QBD-W*.

l_{max}	time (seconds)
1	0.160
2	1.142
3	10.262
4	57.915
5	143.828

Table 4: *QBD-W* run times for different l_{max}.

the returned results. In fact, the results of *QBD-W* consistently outperform those of *QBD-MI*, *QBD-TFIDF*, and *QBD-YAHOO* across all values of k and for all different annotator agreement levels. This higher user satisfaction is a direct result of the "novelty" of the resulted documents compared to the initial query-document. The results of *QBD-MI*, *QBD-TFIDF* (and in a lesser degree *QBD-YAHOO*) tend to contain documents that discuss the same topic as the query-document, and tend to repeat the same points of view. In contrast, the results by *QBD-W* tend to highlight some other, unexpected aspect of the same topic.

Performance: The runtime overheads of our techniques are modest. On a 2.4GHz AMD Opetron processor, both *QBD-TFIDF* and *QBD-MI* require under 300 msecs, of which bulk of the computation time is spent in part-of-speech tagging the document. Access to *QBD-YAHOO* web service takes close to 330 msecs including network latency. Run times for running RelevanceRank algorithm on the wikipedia graph for a typical document as l_{max} is varied are displayed in Table 4. Recall that, for the purpose of computing RelevanceRank scores, parts of G_w at distance greater than l_{max} from the seed set can be ignored. As l_{max} increases, the size of subgraph over which RelevanceRank needs to be computed increases drastically (due to heavy interlinking activity in wikipedia), leading to higher running times. Setting $l_{max} = 2$ or $l_{max} = 3$ works well in practice. Experiments reported in the previous subsection used $l_{max} = 3$.

7. CONCLUSIONS

We have presented techniques to extract candidate phrases from a document in order to utilize them as seed queries to search engines and in particular to BlogScope, a search engine for blogs. We have presented an evaluation using Amazon's MTurk service demonstrating that our retrieval results are of high quality as judged by independent annotators. We believe that the problem of cross referencing documents from different sources is going to become highly significant as the information produced online by services and individuals continues to grow. These features are implemented in BlogScope [4], and we will soon make them available through `www.blogscope.net`.

8. REFERENCES

[1] Aalbersberg, I. Incremental Relevance Feedback. In *SIGIR*, 1992.

[2] Bansal, N., Chiang, F., Koudas, N., Tompa, F. W. Seeking Stable Clusters in the Blogosphere. In *VLDB*, 2007.

[3] Bansal, N., Koudas, N. BlogScope: A System for Online Analysis of High Volume Text Streams In *VLDB*, 2007.

[4] BlogScope http://www.blogscope.net/about/

[5] Church, K. W., Hanks, P. Word Association Norms, Mutual Information and Lexicography. In *ACL*, 1989.

[6] Chandel, A., Hassanzadeh, O., Koudas, N., Sadoghi, M. Srivastava., D. Benchmarking Declarative Approximate Selection Predicates. In *SIGMOD*, 2007.

[7] Crestani, F. Application of Spreading Activation Techniques in Information Retrieval. In *Artificial Intelligence Review*, 1997.

[8] Cucerzan, S. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*, 2007.

[9] Dagan, I., Church K. *Termright: Identifying and Translating Technical Terminology* In *ANLP* 1994.

[10] Efthmiadis, E. Query Expansion. In *Annual Review of Information Science and Technology*, 31:121-187, 1996.

[11] Suchanek, M. F., Kasneci, G., Weikum, G. Yago: a core of semantic knowledge. In *WWW*, 2007.

[12] Fagan, J. Automatic Phrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods. In *SIGIR*, 1987.

[13] Feller, W. *An Introduction to Probability Theory and Its Applications*, Wiley, 1968.

[14] Frantzi, K. T. Incorporating Context Information for the Extraction of Terms. In *ACL*, 1997.

[15] Gravano, L., Ipeirotis, P., Koudas, N., Srivastava, D. Text Joins for Data Cleasing and Integration in an RDBMS. In *WWW*, 2003.

[16] Gyongyi, Z., Garcia-Molina, H., Petersen, J. Combating Web Spam with TrustRank. In *VLDB*, 2004.

[17] Harman, D. Relevance Feedback Revisited. In *SIGIR*, 1992.

[18] Haveliwala, T. Topic-Sensitive PageRank. In *WWW*, 2002.

[19] Ide, E. New Experiments in Relevance Feedback. In *The SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall, 1971.

[20] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady* 1966.

[21] MacDonald, C., He, B., Plachouras, V., Ounis, I. University of Glasgow at TREC 2005: Experiments in Terabyte and Enterprise Tracks with Terrier. In *TREC*, 2005.

[22] Manning, C., Schutze, H. *Foundations and Statistical Natural Language Processing*, MIT Press, 1989.

[23] Medelyan, O. Computing Lexical Chains with Graph Clustering In *ACL* 2007.

[24] Medelyan, O., Witten, I. Thesaurus Based Automatic Keyphrase Indexing In *JDCL* 2006.

[25] Mitra, M., Buckley, C., Singhal, A., Cardie, C. An Analysis of Statistical and Sytactic Phrases. In *RIAO Conference*, 1997.

[26] Mittendorf, E., Mateev, B., Schauble, P. Using the Co-occurrence of Words for Retrieval Weighting. In *Information Retrieval*, 3(3): 243-251, 2000.

[27] Amazon Mechanical Turk. *http://www.mturk.com*

[28] Pantel, P., Lin, D. *A statistical corpus based term extractor* Lecture notes in AI, 2001, Springer-Verlag

[29] Part-of-speech tagging. *http://en.wikipedia.org/wiki/Part-of-speech_tagging*

[30] Rocchio, J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*, Prentice-Hall, 1971.

[31] Salton, G. and McGill, M. J. *Introduction to modern information retrieval*. McGraw-Hill, 1983.

[32] Spink, A., Jansen, B., Ozmultu, H. Use of Query Reformulation and Relevance Feedback by Excite Users. In *Internet Research: Electronic Networking Applications and Policy*, 2000.

[33] Tomokiyo, T., Hurst, M. *A Language Model Approach to Keyphrase Extraction* In *ACL* 2003

[34] Turney, P. D. Learning Algorithms for Keyphrase Extraction. In *Information Retrieval*, 2000.

[35] Vechtomova, O., Karamuftuoglu, M. Approaches to High Accuracy Retrieval: Phrase-Based Search Experiments in the HARD Track. In *TREC*, 2004.

[36] Witten, I., Paynter, G., Frank, E., Gutwin, C., Manning, G. KEA: Practical Automatic Keyphrase Extraction. In *ACM DL* 1999.

[37] Yahoo Term Extraction Web Service. *http://developer.yahoo.com/search/content/V1/termExtraction.html*

[38] Zaragoza, H., Rode, H., Mika, P., Jordi, A., Ciaramita, M., Attardi, G. Ranking Very Many Typed Entities on Wikipedia. In *CIKM*, 2007.

[39] The Future of Social Networking: Understanding Market Stratigic and Technology developments. Datamonitor, 2007.