Similar Document Search and Recommendation

Vidhya Govindaraju, Krishnan Ramanathan

HP Laboratories
HPL-2011-150

Keyword(s):

Key phrase extraction; recommendation system; similarity search

Abstract:

Query formulation is one of the most difficult aspects of search, especially for a novice user. We propose a new search interaction where the user searches with a reference document and the system learns from the user inputs over a period of time to "push" relevant and new content without additional user interaction. Our method is based on identifying key phrases from the input document. The key phrases are used to query a search engine and the results are evaluated for similarity to the original document. By caching documents received from a user over a period of time, a user profile is built. The profile is then used to provide recommendations to the user.

External Posting Date: September 21, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: September 21, 2011 [Fulltext]

Similar Document Search and Recommendation

Vidhya Govindaraju
HP Labs, Bangalore, India
vidhya.govindaraju@hp.com

Krishnan Ramanthan
HP Labs, Bangalore, India
krishnan_ramanathan@hp.com

***Abstract*—Query formulation is one of the most difficult aspects of search, especially for a novice user. We propose a new search interaction where the user searches with a reference document and the system learns from the user inputs over a period of time to "push" relevant and new content without additional user interaction. Our method is based on identifying key phrases from the input document. The key phrases are used to query a search engine and the results are evaluated for similarity to the original document. By caching documents received from a user over a period of time, a user profile is built. The profile is then used to provide recommendations to the user.**
Evaluations show that this method has a good precision in finding documents of interest to the user. Also our key phrase extraction method has good recall in retrieving the input document. Additional experiments reveal that our recommendation system is of help in exploring documents of interest to the user.

***Index Terms*— Key phrase extraction, recommendation system, similarity search.**

I. Introduction

In spite of the ubiquity of search engines, navigating information spaces remains a complex affair. Traditional search operates by matching a search query to (pre-processed) document representations. While current search algorithms perform reasonably well when the goal is navigation and known item search, they are not well suited when the goal is more exploratory and persistent in nature (e.g. the user is looking to learn a new topic). The user's ability in finding relevant information depends on his ability to frame good queries. However, query formulation is harder when the user is unfamiliar with the topic. There is also very little support on the web for stating persistent interest; this is necessary for facilitating ongoing learning. These long term interests are often stated in the form of short queries for which an engine can provide alerts [11], however ongoing maintenance of alerts is a problem.

Often, users have a set of documents obtained through browsing or from their social network (via emails, recommendations etc). These documents could serve as a good starting point for further search and recommendations. These documents are highly reflective of the user interests, yet they are hardly used in fulfilling ongoing information needs. Today, it is the responsibility of the user to identify salient keywords from the specific document of interest and use them in a search query.

In this paper, we propose an interaction paradigm whereby a user can provide to a relevant document and ask the system to retrieve similar documents without having to formulate search queries. For example, we would like the system to take as input the PDF version of John Hopcroft's talk on "Future directions in Computer science" (www.cs.cornell.edu/jeh/**China**%202007.ppt) and output Ed Lazowska's talk titled "Computer Science-past, present and future" (lazowska.cs.washington.edu/fcrc/Lazowska.FCRC.pdf) as a similar document. Since the user is likely to be interested in similar documents that get created at a later point in time, it would be useful for a system to scout for similar documents on a continuous basis and send it to the user whenever they become available.

There are three main goals of the system. They include

1. Fetching documents similar to an input document.
2. Learn user interests periodically and recommend documents covering multiple user interests.
3. Provide enough content exploration via result diversification.

Key phrases are often used as a brief summary of documents. Hence they could prove useful for retrieving and recommending similar documents. Since manual key phrase extraction is time-consuming, automatic extraction becomes an important task. We describe a novel key phrase extraction method to extract key terms in a document and use them in a query to find similar documents. The aim of this is to find key phrases that best describe the context of the document.

The amount of content on the web is increasing with new

Manuscript received January 1, 2011; revised June 1, 2011; accepted July 1, 2011.

articles, documents, blogs etc being posted every day. Users face the problem of finding documents in their area of interest. There is a lot of support in the web for finding new and relevant information. Web based recommender systems help in choosing the right documents for the user. Often such systems suffer from the problem of data sparsity and hence produce redundant results. We develop a personalized recommendation framework for recommending documents based on the past user requests to the system. A user profile is built from the past requests and then used to source and recommend documents to the user on an ongoing basis. We feel that such a push based document system will be highly valuable in finding content from the web without querying for it.

We extend our solution of finding similar and relevant content to result diversification. A document has multiple modalities and providing the user with similar content in all these dimensions becomes an essential component in this scenario. We study the problem of diversifying search results and present ways to maximize relevance and diversity of search results.

There are a number of applications where this kind of technology can be useful. For example, in e-discovery, a patent attorney looking for relevant documents among millions of documents can identify one relevant document and request the search system for similar ones. In an online video application, a user can mark videos as interesting and request retrieval of similar videos. Finally, in an exploratory search scenario, users might find the results useful even if they are not very similar to the input document. We would like to stress that in this work, our motivation is not to detect duplicate web pages or documents.

The remainder of this paper is organized as follows. In section 2 we survey related work on similar document search, keyword extraction, results diversification and user profiling. In section 3 we describe the architecture of the system. Section 4 discusses the system components and algorithms in greater detail. In section 5, we describe the different fronts on which we evaluated our system. Section 6 concludes the paper.

II. RELATED WORK

A. *Similar Document Search*

Similarity search has recently become a field of active research [7] [8]. Despite this, there are very few systems that use similarity search to facilitate user interaction.

One of the earliest approaches was the "Similar pages" or "More like this" [10, 11] link provided by search engines for search results. In [9] related article search in Pubmed through citation links in the database is presented. The user study reveals that such a system which helps in exploring new and relevant information is a useful feature and it becomes an integral part of user's interaction with Pubmed. A direct way of finding similar text that is conceptually related to the input document is presented by Yang et.al. [7]. They have built a system for finding similar articles in BlogScope. They have developed a system of cross-referencing information created by different users. There is a large amount of related work on retrieving similar images. For example, Flickner [12] developed a system for querying with images to get similar images and videos.

B. *Keyword and Key phrase Extraction*

The simplest approach to key phrase extraction is taking top n most frequent n-grams in a document [5]. A method for extracting keywords based on frequency and co-occurrence phenomenon is presented in [1]. In [7] key phrases are extracted using part of speech tagger. All noun phrases are extracted as key phrases.

Yahoo Phrase Extractor [3] takes a text snippet and returns key terms in the text. In [15], the task of key phrase discovery is accomplished using suffix arrays or suffix tree structures. They have also presented the benefits of using key phrases as a feature in natural language processing. The authors have used key phrases extracted from web pages in clustering web search results.

Kea algorithm [6] uses the Naïve Bayes machine learning algorithm for training a classifier with user generated key phrases for sample documents. The trained set is then used to extract key phrases from other documents.

Extracting key terms from noisy documents by exploiting the graph of semantic relationships between terms in the document is explained in [4]. This method is close to ours except that they exploit the Wikipedia information to filter redundant phrases. In [14], clustering based unsupervised key phrase extraction algorithm is presented.

Since most key phrase extraction methods generate a large number of key phrases, some form of ranking is used to select key phrases [2]. Most key phrase extraction algorithms are based on TFIDF for ranking key phrases [5].

C. *Diversifying search Results*

Documents have multiple themes associated with it. Our hypothesis is that in the process of finding a document that is similar to the input document, users want variety and coverage of different themes that the input document covers. Hence diversifying search results to cover these multiple interpretations becomes important. Diversification can be achieved in two ways: framing multiple queries that are intrinsically diversified or clustering results so as to achieve diversification. Agrawal et.al. [13] employed a

greedy algorithm that minimizes user dissatisfaction in a web search scenario. This method considers the popularity of the category while diversification.

Clustering of words [14] will help in framing queries that represent various themes in a document. Clustering words that are semantically similar is done based on known ontologies. Also mutual information between words could be used as a factor to classify words into different clusters.

The traditional method of clustering documents considers a document as a vector of words and distance between two documents is found by taking the cosine similarity between them. This is then used in a hierarchical clustering algorithm to get document clusters. But this method leads to high dimensionality and the computational costs are often huge. Using key phrases as document features for clustering is discussed in [15].

D. *Recommending documents based on user profile*

Web based recommender systems are primarily based upon Collaborative Filtering (CF) techniques which filters information based on user preferences. It is based on measuring the similarity of users or items or both [21]. Though the user based CF has a lot of commercial applications, when sorted for recommending documents this suffers from the problem of data sparsity and noise. Zhou [17] has used co-citation graph, author-document relationship and document-venue relationship in an item based CF for recommending documents. They implemented a single low dimensional embedding of documents that capture the similarities between them. A semi-supervised learning on this graph was used to develop a recommendation system. In [18], a content based recommendation system for Citeseer database is proposed. They classify the documents in Citeseer into predefined set of concepts which they use to build a user profile and recommend documents accordingly.

Query specific recommendation depending on standing interests is proposed in [19]. Xu et. al [20] proposes a personalized method for recommending documents based on eye tracking of keywords in the document.

III. SYSTEM OVERVIEW

In this section, we present an overview of our system architecture and designed a solution to the problem of recommending relevant documents based on a set of input documents.

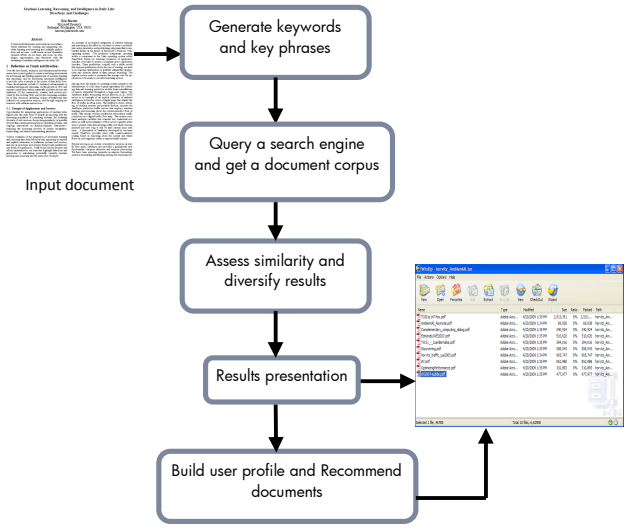

Figure 1. System Overview

When a user queries our system with a document, our system generates keywords and key phrases from the document and uses them in a search query to get an initial set of documents from the web. It ranks these documents based on the similarity to the input document and diversifies the results so as to cover all the themes in a document. It then presents the top ranked similar documents to the user. It builds a profile with the documents received from a user and exploits them in sending recommendations. An overview of the system architecture is shown in figure 1.

The first step in our solution is to generate a set of keywords and key phrases from the input document. A document can be viewed as a bag of words. Identifying the most important words can help in framing the right queries for searching similar documents. The keywords and key phrases in the documents appear often in document titles, paragraph titles, and important sentences, often associated with more meaningful terms. We exploit this feature in finding keywords and key phrases in the document. Since key phrases are more descriptive than keywords in explaining the context of the document, we use them in framing a search query. We also add the most important keywords that may not have a co-occurrence feature to the list while framing a search query.

We achieve search result diversification by framing multiple queries representing various dimensions of a document. We cluster the key phrases into different sets and frame the queries for each cluster using the key phrases in it. We query the search engine (e.g. Google scholar) with these clusters and get an initial set of documents. Since the queries are intrinsically diversified, the initial corpus contains documents diversified on various topics. For each document retrieved, we assess

their similarity with the input document. We return most similar documents to the user.

We extend our solution to develop a recommendation framework based on content similarity. We build the user profile with the keywords and key phrases of the documents that are sent to the system by the user. We also use the author information in the input documents to recommend recent documents published by the authors to the user. This is highly useful in a research scenario to find other content that is posted or created by the same author whose documents the users are interested in. Also in a document search application, finding new publications from the top publishers who publish content of interest to the user is also an important feature. Hence we use the profiler data in finding the publishers and extract new and relevant content published by them for recommending to the users. Our evaluations show that such a system is very useful in exploring the web for finding relevant information.

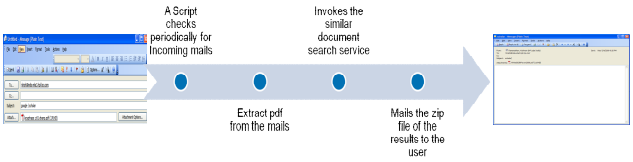

Figure 2. User Interface Overview

The service is exposed as a cloud service. On the server side, it is implemented as a shell script in Linux which runs every minute and checks the incoming mails and parses them to get the required information. This script then runs the program for the similar document search and collects the similar documents, compresses them and mails it back to the user (figure 2). Currently, ten relevant documents are mailed to the user.

IV. PROPOSED SOLUTION

In this section, we describe the algorithms used in our system.

A. *Key phrase Extraction and Ranking*

In our method, we model the document as a graph of words in which the important words in the document tend to co-occur with other important words. We exploit this feature in finding the keywords from the document. We extend the algorithm in [1] to generate keywords and key phrases for a document.

We first construct a graph where the nodes are high frequency words in the document. The weight of a word is taken as the document frequency of the word (excluding paragraphs and document titles) multiplied by its weight. The weight of a word is measured by the frequency of the word in paragraph and document titles. If it does not appear in paragraph and document titles, the weight is taken as one.

We create edges between those nodes of the graph if words associated with the nodes co-occur in the same sentence. The edge weight is the minimum of the word weights assigned to words in the previous step. We add to the graph those words that co-occur with the high frequency words in any sentence. We then find the maximally connected components in the graph. Each maximally connected component is called a concept. We use the concept graph (C_G) for finding the keywords and key phrases.

We find all 2-3 gram words in the document that does not contain a stop word in the middle. For each phrase, we test whether the words in the phrase are part of a concept. We find the rank of the phrase as follows.

$$\text{Rank of a phrase} = \frac{\text{Number of words in phrase which are present in the concept graph} * \text{Frequency of a phrase}}{\text{Number of words in the phrase}} \quad (1)$$

We select phrases with higher rank as key phrases. Sometimes, key phrases generated by our method are permutations of each other. These key phrases are further filtered so as to avoid redundancy. While selecting a new key phrase, only those that contain a new keyword (compared to previous key phrases) are chosen.

From our initial experiments, we found that adding a few keywords that are very important in the document, to the seed query can improve the precision of the results drastically. This is because certain words may not have a co-occurrence feature. So for adding the keywords that are very important to the query we use the weight of the keywords. We sort the keywords in the descending order of their weight. We find the point at which there is steep decrease in weight. All the keywords before this point are taken for query formulation.

B. *Querying the search engine and assessing similarity*

A query is formulated using the key phrases and important keywords and is used to query the search engine to get a set of similar documents.

The retrieved documents are ranked based on the similarity to the input document. There are a number of ways in which similarity could be assessed. In our method, keywords and key phrases are extracted from the retrieved document. Jacquard similarity measure is found between the keywords and key phrases of input and retrieved document using the equation

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

Here A and B are the key phrases extracted from the input and retrieved documents. Documents with higher similarity score are sent to the user.

C. *Diversifying search results*

A document could be addressing multiple themes. For instance, a research paper on mobile security could be addressing issues with mobility and security. Diversifying search results so as to cover all the major themes of a document will increase user satisfaction. This can be achieved by framing multiple queries one for each context.

D. *Framing multiple queries*

For framing multiple queries, the phrases and important keywords extracted above are clustered based on their semantic relationships into different groups. We use the Normalized Google Distance (NGD) [16] as a measure to find semantic relationships between words. NGD uses Google page counts of words and phrases to find the relative distance between them. All the key phrases and important keywords in the input document are clustered based on the NGD between them. These clusters are used to frame multiple queries so as to cover all the context of an input document. A sample cluster of key phrases and keywords for the research paper in reference [7] is listed in figure 3.

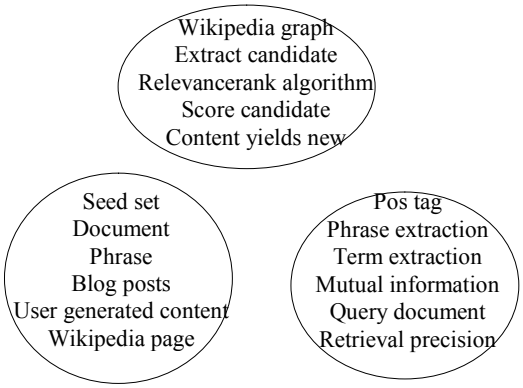

Figure 3. Key phrase clusters for reference [7]

E. *Algorithm for clustering key phrases*

Input: Number of required clusters N, Array of key phrases

Output: Clustered key phrases

Steps:

1. Find Normalized Google Distance (NGD) between every pair of key phrases and important keywords.

2. Use the NGD between key phrases in a hierarchical agglomerative clustering algorithm to cluster the key phrases. Compute the cluster centroid. This step is used to find cluster centriods for seeding the k-means clustering algorithm.

3. With these cluster centroids, use the NGD between key phrases in a k-means clustering algorithm to get clusters of key phrases.

F. *Building User Profiles*

A profile is a description of user interests. A push based data delivery system requires a profile to be built for every user interacting with the system. The user interacts with our system by sending a document of significance to him and requesting for similar documents. The input document serves as a tool to predict user interest. We aim at building a user profile by implicitly predicting user interest from user interactions.

We add the keywords and key phrases from the input document to the profile. The profile thus built for a user contains terms that broadly specify his interest. When keywords and key phrases collected over a period of time repeat in the profile or have a close semantic relationship between them, it represents consistent user interests. The recommendation algorithm exploits this feature in getting valuable recommendations for the user.

We also add other metadata of the input document such as author name to the profile. This specifies the list of authors whose documents the user has read. The user profile thus built is used in a recommendation system to push relevant content to the user.

G. *Recommendation System*

The system is used to recommend more relevant and recent documents to the user based on his profile. The recommendation system uses multiple approaches to improve relevance.

a) Recommend documents based on user profile

b) Recommend new documents of favorite authors

c) Recommend new documents from recent conferences or journals.

Recommend documents based on user profile

The profile built for a user implicitly using the documents received from the user, is used to select documents for recommendation. The profile has a list of keywords and key phrases from the input documents and author names of the documents. All the key phrases and keywords in the profiler are clustered (using the algorithm in Section 4.3.1.1) into different clusters. Since the clusters are framed from keywords and key phrases taken from multiple documents, they capture interrelationships between various topics. One cluster may contain key phrases from different documents if they have a close semantic distance.

These clusters are then used to frame multiple queries. An initial set of documents are retrieved using Google Scholar as the search engine for each of these queries. For each

document retrieved we extract the keywords and key phrases.

Each document is ranked based on three parameters:

S_P - Similarity with the profiler

D_P - Days since the document was published

N - Number of queries that retrieved the same document.

$$\text{Rank} = (S_P * \alpha + (1/ D_P) * \beta) * N \quad (3)$$

The similarity score is the strength of user interest in the document. Similarity with the profiler is computed as follows.

$$\text{Profile Similarity Score, } S_P = \frac{|A \cap B|}{|A \cup B|} \quad (4)$$

Here, A is the set of key phrases in the document and B is the set of key phrases in the user profile.

The second parameter is used to filter out old documents and ensure recency. For experimental purposes, we used $\alpha = 1$ and $\beta = 4$. The documents are sorted based on the final rank and top ranked documents are recommended to the user.

Recommend new documents of favorite authors

The profiler together with the list of keywords and key phrases of the documents received from a user, has a list of authors of each document. We assume that this list of authors as favorite authors for the user. The number of documents of a particular author, that a user has read is taken as the authority of an author. We extract papers for each author by adding the string author:<name> to the query in the search engine. Also we find all the co-authors from the input list and retrieve papers for each double author. We extract keywords and key phrases for each document retrieved.

Each document is thus ranked on four parameters

S_P - Similarity with the profiler

D_P -Days since the document was published

N_A - Number of favorite authors who wrote this document

A_W - Authority of the authors who wrote the document

$$\text{Rank} = S_P * \alpha + (1/ D_P) * \beta + N_A * A_W * \gamma \quad (5)$$

Author names may be misleading when the name is shared by more the one person who publishes content in different fields. To prevent such errors, we compute the similarity of the document with the profiler. Similarity with the profiler is computed as in Equation (4). The second parameter in the above equation is used to filter old documents. The third parameter increases the weight of a document according to the number of favorite authors who wrote the document. For experimental purposes we used $\alpha = 4$, $\beta = 4$ and $\gamma = 2$.

Recommend new documents from recent conferences or journals

We exploit the profiler key phrases in fetching a list of conference and journal names that publish content of interest to the user. This helps in alerting the user with relevant documents from recent conferences.

All the key phrases and keywords in the user profile are clustered (using the algorithm in Section 4.3.1.1) into different clusters. These clusters are then used to get a list of conference and journal names from the search engine. For each clustered dataset, we get publisher details of documents from the search engine (e.g. in Google Scholar using the Bibtex output) and add them to the publishers list. We consider publishers with higher frequency from the results list and retrieve recent documents published by them. This is done by querying the conference name in the search engine and setting the recent preference to the current year. For each conference, recent documents that were published by them are obtained.

Each document is ranked based on two parameters

S_P - Similarity with the profiler

D_P - Days since the document was published

$$\text{Rank} = S_P * \alpha + (1/ D_P) * \beta \quad (6)$$

Similarity with the profiler is computed as in Equation (4). The second parameter is used to filter old documents. For experimental purposes, we used $\alpha = 1$ and $\beta = 4$. We used these values because more likely the documents retrieved with the conference names tend to be relevant and the major parameter here is the published date (since users desire documents that are recommended to be more recent).

V. EXPERIMENTAL EVALUATION

Our experimental evaluation is designed to answer the following questions

1. Are the key phrases obtained by our method sufficiently discriminative?

2. Are the similar documents retrieved by the system of good precision?

3. How good are the recommendations made by our recommender?

A. *Evaluation of Key phrases*

We first evaluated the usefulness of using key phrases in representing the context of a document. This was done because the hypothesis we use for retrieving similar documents is that they are clustered close to the input document. We extracted keywords and key phrases from a document published in the web. We used the keywords and key phrases separately as a query to find the rank at which

the input document is retrieved. Results (in figure 4) show that key phrases perform better than keywords in retrieving documents at higher rank.

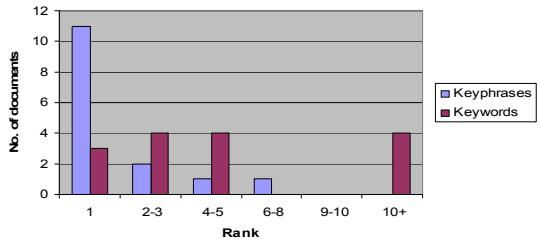

Figure 4. Comparison of keywords vs. key phrases in fetching the input document

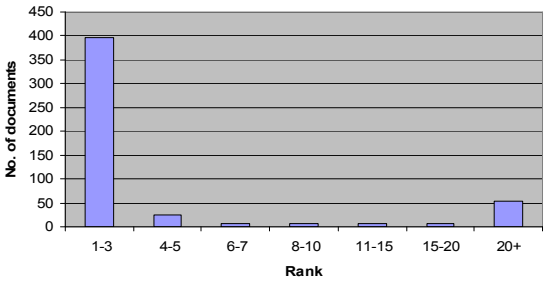

Figure 5. Performance of key phrases in fetching the input document

We evaluated whether the key phrases extracted for a document is able to retrieve the same document back in top results in Google scholar. An automated test run for 500 documents shows that 80% of the documents were retrieved at top ranks by querying with the key phrases. Results are shown in figure 5.

We next compared our method of finding key phrases against existing methods. We compared our key phrase extraction method with the Yahoo Phrase extractor and manually extracted key phrases. Yahoo! Phrase Extractor is on online tool for extracting phrases. It takes a text document as input and returns a list of key phrases for the document.

We conducted a study in which the users were presented a list of key phrases for each document in the test data set. The list comprised of the key phrases extracted by our method and Yahoo Phrase Extractor. The users were asked to rate the relevancy of each key phrase (0, if it is irrelevant and 1, if relevant). We present the comparison using the measure of precision which is calculated as in Equation 7.

$$\text{Precision} = \frac{\text{No of relevant key phrases}}{\text{Total no. of key phrases}} \quad (7)$$

Table 1 summarizes the performance of our method and Yahoo Phrase Extractor for the test data set.

TABLE I

Performance of different key phrase extraction methods

Method	Precision
Yahoo! Phrase Extractor	0.478
Our key phrase extraction algorithm	0.758

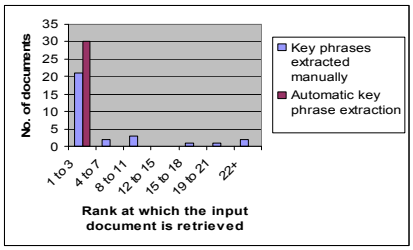

Figure 6. Comparison of automatic key phrase extraction against manually selected key phrases

A study was done in which users were asked to find key phrases in 30 documents. Key phrases were also extracted by our method. Using these phrases a search string was framed and the rank at which the input document is retrieved back is found in Google. Figure 6 shows that our key phrase extraction algorithm performs better than manually extracted key phrases in retrieving the input document at higher rank.

B. *Evaluation of Similar Documents*

In this section we evaluate the relevance of similar documents that are retrieved by our method. We conducted an evaluation of the system with 10 users by exposing the solution as a cloud service. The users were primarily research scholars.

To compare the retrieval quality of the system, we computed the following commonly used measure:

1 Precision at K: Prec@K(q) is the fraction of relevant documents within the top K results for a query q. This needs a binary classification of documents.

2 Mean Average Precision : It returns a single value for each method of ranking and is computed as follows

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\Pr ecision(R_{jk}) \quad (8)$$

Q is a set of queries, m_j is the number of documents on which precision is computed and is chosen as 20. MAP takes the position of the results in the ranking into account and is based on binary relevance classification.

3 DCG at K: The Discounted Cumulative Gain explicitly takes the ranking of first K results into account. Hence it rewards highly relevant results less than less relevant results. It is computed as follows :

$$DCG@K(q) = \sum_{j=1}^{K} (2^{r(j)} - 1) / \log(1 + j) \quad (9)$$

Here, r(j) is an integer for the relevance rating given to the result at position j for the query q and is taken on a scale of 1-3..

We first evaluated the efficiency of key phrases in finding similar documents. Here, all the key phrases are given as one long query. In this experiment the user sends a document to the service and is sent a set of documents similar to the input document. The user is asked to rate each document on a scale of 1-5 based on his interest in the document.

We chose ten documents from varied topics for evaluating the relevancy of similar documents. For each input document, we retrieved 20 similar documents. We used the above specified measures for evaluation. Figure 7 shows the precision graph and Figure 8 shows the DCG graph. Results are summarized in the Table 2.

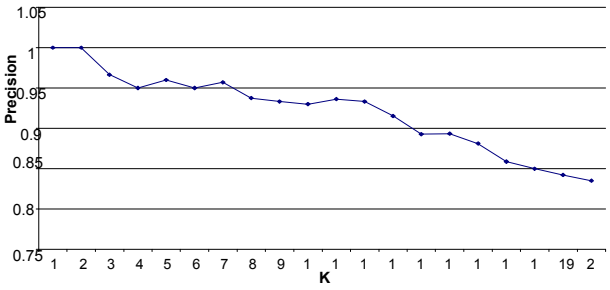

Figure 7. Precision at K

TABLE II

Evaluation of similar document search results

Precision @ 10	0.93
Precision @ 20	0.835
Mean Average Precision (MAP)	0.90

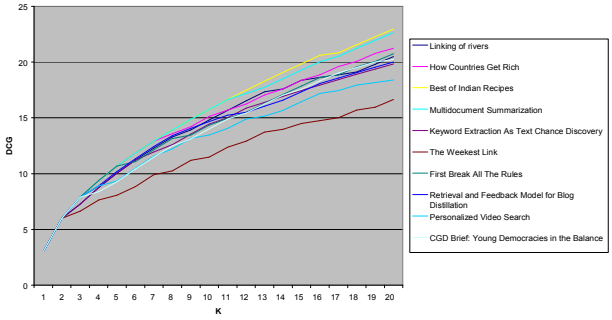

Figure 8. DCG at K

Also, we evaluated the importance of diversification in search results. We will compare the similar documents that are retrieved by the system with and without clustering of key phrases. For this, we took ten documents from the users and sent two lists of 10 documents for each input document. The documents considered were primarily research papers that were much focused. For each document sent, the user rated the relevancy of it on a scale of 1-5. We use the standard measure of precision and DCG for comparison. Figure 9 shows the DCG graph. Table 3 summarizes the performance of our method.

TABLE III

Precision of similar documents retrieved with and without explicit result diversification

Method	Precision @10
Similar Documents retrieved with result diversification	0.64
Similar Documents retrieved without result diversification	0.65

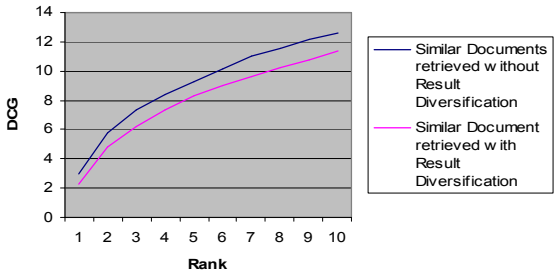

Figure 9. DCG graph of similar documents retrieved with and without explicit result diversification

C. *Evaluation of Recommended Documents*

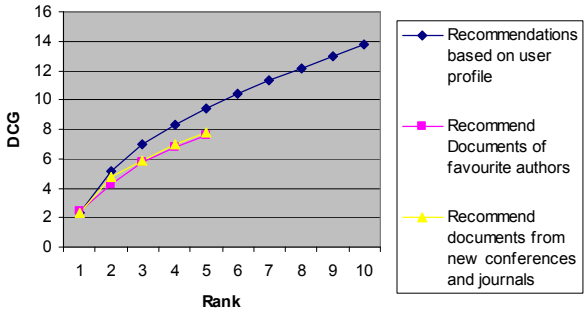

Figure 10. DCG graph for recommended documents

For evaluating the recommender system, the users are sent a series of recommended documents based on the documents submitted by them to the system. The user feedbacks are collected for each recommended document on a relevancy scale of 1-5. The DCG graph (Figure 10) shows that our recommender system based on user profiles performs well in selecting the right documents for recommendation. Table 4 summarizes the performance of our individual recommendation modules. This study was done with 10 users, however we could obtain more than five relevant documents for only 7 users (because of subscription requirements), hence results are reported for only 7 users.

TABLE IV

Performance of Recommender System

Recommendation Module	Performance	
Recommend documents based on user profile	Precision @ 10	0.89
Recommend new documents of favorite authors	Precision @ 5	0.51
Recommend new documents from recent conferences or journals	Precision @ 5	0.65

D. *Discussion of Experimental results*

The evaluation of extracted key phrases reveals that the key phrase extraction algorithm performs better than some existing key phrase extraction methods. It is also noted that key phrases performs better than keywords in describing the context of a document. Evaluation of similar documents shows that our method helps in exploring the web in finding documents of interest to the user. However, it is noted that the diversifying search results does not improve the user satisfaction. This could be because the user may be interested in only the broader theme of a document and not all the themes. Clustering has taken the results to a different document space that is more precise to the individual themes in a document.

From the evaluation of our recommender system we infer that such a system can form a useful component of the user's information exploration in the web. Also it is evident that the recommendation based on user profile has fetched documents of interest to the user.

VI. CONCLUSION

In this paper, we propose a novel retrieval approach for document similarity search. We have formulated a method to get similar documents based on the concept of the input document by extracting the relevant keywords and key phrases from the document. The experimental results have shown favorable performance of the proposed approach. We have also built a personalized document recommendation system based on the user profile created implicitly with the inputs received from the user.

In future, we will aim to extend the similarity search and recommendations to web pages, video content and to cross-lingual similarity search where information in one language could be used to find similar information in other languages

REFERENCES

[1] Z. Zhang, H. Cheng, Keyword extracting as text chance discovery, IEEE Fuzzy systems and knowledge discovery conference (FSKD), 2007.

[2] Xin Jiang, Yunhua Hu, Hang Li, A Ranking Approach to Key phrase Extraction, Proc. SIGIR 2009.

[3] Yahoo Phrase Extractor, http://developer.yahoo.com/search /content/V1/termExtraction.html

[4] M. Grineva, M. Grinev, and D. Lizorkin. 2009. Extracting key terms from noisy and multi-theme documents, Proc. WWW 2009, pages 661–670.

[5] Lee, J.W. and Baik, D.K., A model for extracting keywords of document using term frequency and distribution, Lecture notes in computer science, Springer, Pg. 437—440, 2004.

[6] I. Witten, G. Paynter, E. Frank, C. Gutwin, and C. Nevill-Manning, Kea: Practical automatic key phrase extraction, Proceedings of the 4th ACM conference on Digital Libraries.

[7] Yang Y., Bansal, N., Wisam, D., Panagiotis, I., Koudas, N., Dimitris, P., Query by Document, WSDM '09.

[8] Xiaojun Wan, Jianwu Yang, Jianguo Xiao, Document Similarity Search based on Manifold Ranking of TextTiles, AIRS 2006, LNCS 4182, pp. 14 – 25, 2006.

[9] Jimmy Lin, Michael DiCuccio, Vahan Grigoryan, W.John Wilbur, Navigating information spaces: A case study of related article search in PubMed,

Information Processing and Management, 2008, Elsevier

[10] Google similar pages, googleguide.com/similar_pages.html

[11] Google alerts, http://www.google.com/alerts

[12] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yonker. Query by image and video content: The qbic system. Computer, 28:23–32, 1995.

[13] Rakesh, A., Sreenivas, G., Alan, H., Samuel, I., Diversifying search results, WSDM '09.

[14] Z. Liu, P. Li, Y. Zheng and M. Sun, Clustering to Find Exemplar Terms for Keyphrase Extraction, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.

[15] Dell, Z., Yisheng, D., Semantic, Hierarchical and Online Clustering of Web Search Results, AP Web 2004, LNCS 3007, pp 69-78, 2004.

[16] Rudi L. Cilibrasi and Paul M. B. Vitanyi, The google similarity distance, IEEE Transactions on Knowledge and Data Engineering, 19(3):370–383, 2007.

[17] D Zhou, S Zhu, K Yu, X Song, BL Tseng, H Zha, Learning Multiple Graphs for Document Recommendation, Proc. WWW 2008.

[18] Kannan, C., Susan, G., Praveen, L., HP Luong, Concept-Based Document Recommendations for Citeseer Authors, Lecture Notes in Computer Science, 2008 – Springer

[19] B. Yang, G. Jeh, Retroactive Answering of Search Queries, Proc. WWW 2006.

[20] S Xu, H Jiang, FCM Lau, Personalized online document, image and video recommendation via commodity eye-tracking, Proc. RecSys 2008

[21] Fei Wang , Sheng Ma , Liuzhong Yang , Tao Li, Recommendation on Item Graphs, Proceedings of the Sixth International Conference on Data Mining, p.1119-1123