

SPEAR: SEARCH PERSONALIZATION WITH EDITABLE PROFILES

by

Binyam A. Zemedu, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
December 2013

Committee Members:

Byron J. Gao, Chair

Anne H.H. Ngu

Yijuan Lu

COPYRIGHT

by

Binyam A. Zemedu

2013

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

This work is protected by the Copyright Laws of the United States (Public Law 94-553, section 107). Consistent with fair use as defined in the Copyright Laws, brief quotations from this material are allowed with proper acknowledgment. Use of this material for financial gain without the author's express written permission is not allowed.

Duplication Permission

As the copyright holder of this work I, Binyam A. Zemedu, authorize duplication of this work, in whole or in part, for educational or scholarly purposes only.

DEDICATION

I dedicate this to my wife, Helen Afework, my mother, Abebech Tekele and my father, Abebe Zemedu.

ACKNOWLEDGEMENTS

Foremost, I would like to thank Dr. Byron J. Gao, who inspired and motivated me to carry out and complete the thesis. Dr. Gao has always made himself available and has been a source of constructive feedback. I would also like to thank Dr. Anne H.H. Ngu and Dr. Yijuan Lu for their involvement in the thesis committee during the thesis process. I acknowledge my manager, Rick Aberle, for his support throughout my Master's degree studies. I would like to thank Teodros Zemedie, Brian Jackson and Christina Jackson for their participation in proofreading the thesis. I would also like to thank my family and friends for their support and encouragement. Finally, I would like to thank my beautiful wife, Helen Afework, for her support and patience throughout the thesis process. I am grateful to GOD for all the blessings in my life.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF ABBREVIATIONS	x
ABSTRACT	xi
CHAPTER	
1. INTRODUCTION	1
1.1 Information Overload and Search Ambiguity	1
1.2 Search Personalization	2
1.3 The SPEAR Approach	3
2. BACKGROUND	9
2.1 Information Retrieval	9
2.2 Vector Space Model	11
2.3 Information Retrieval System Evaluation	15
2.4 Semantic Similarity	19
3. LITERATURE SURVEY	20
3.1 Search Personalization	20
3.2 Social Search	23
4. THE SPEAR APPROACH	25
4.1 SPEAR Architecture	25
4.1.1 Profile Management Module	27
4.1.2 Social Network Feed Module	32
4.1.3 Query Processing Module	33
4.1.4 Search History Module	33

4.1.5 Personalization Module	33
5. SPEAR IMPLEMENTATION	35
5.1 Database Design	36
5.2 Object Relational Mapping	37
5.3 Web Services	38
5.3.1 Personalization Service	38
5.3.2 CRUD Service	39
5.4 Web Pages	40
6. EVALUATION	41
6.1 Test Cases	43
6.2 Results	45
6.2.1 11-Point Interpolated Average Precision	45
6.2.2 Precision at K	46
6.2.3 Recall at K	47
6.2.4 MAP (Mean Average Precision)	48
6.3 Discussion	49
7. CONCLUSION	50
APPENDIX A. SPEAR INSTALLATION AND ADMINISTRATION	51
REFERENCES	54

LIST OF TABLES

Table	Page
6.1 Test profiles	43
6.2 Test cases	44
6.3 MAP values	48
A.1 SPEAR system requirements	51

LIST OF FIGURES

Figure	Page
1.1 A sample search in SPEAR	4
2.1 Inverted index construction	10
2.2 Precision/recall graph	17
4.1 SPEAR functional modules	26
4.2 Profile page anatomy	30
5.1 SPEAR database entity relationship diagram	36
6.1 11-point interpolated average precision	45
6.2 Precision at K	46
6.3 Recall at K	47
6.4 MAP	48

LIST OF ABBREVIATIONS

Abbreviation	Description
AJAX - Asynchronous JavaScript and XML	A Web development technique for asynchronous Web applications
API - Application Programming Interface	A description of functionality that defines interfaces between software components
IR - Information Retrieval	A computer science discipline that focuses on search techniques for documents of unstructured nature
HTML - Hyper Text Markup Language	A language for describing Web pages
JSON - Java Script Object Notation	An electronic data format popular in Web applications
SQL - Structured Query Language	A programming language to manage data in a relational database
URL - Uniform Resource Locator	A Web address
WCF - Windows Communication Foundation	A set of APIs for building service oriented applications in the .NET framework
XML - Extensible Markup Language	Structured data description language

ABSTRACT

Search personalization is an important technique for improving search performance. Existing approaches work in a *black box*, where users have no clue on how it works and how to customize it. This lack of user control and flexibility can often be inconvenient and counter-productive. In this thesis we propose SPEAR, a *transparent* search personalization framework that enables full user control and manipulation. In SPEAR, a user can own multiple profiles and each can be modified arbitrarily. Profile terms can be manually entered or automatically generated from search history or social network feeds. Furthermore, the terms can be automatically expanded by adding their semantically related derivatives. Negative terms are allowed for specification of negative preferences, which can be very useful in filtering out undesirable results. The in-use profile will help re-rank search results based on how consistent they are with respect to the profile. We implement SPEAR in the context of a Web search using Google Web search API and Facebook Graph API, demonstrating the promise and potential of the approach.

CHAPTER 1

INTRODUCTION

With wide availability and popularity of the Web, the digital information generated by the world is growing at an exponential rate. In 2000 alone, the world produced an estimated 1.5 exabytes¹ of digital information. In 2011, that number grew to a staggering 1.8 zettabytes² of new digital information produced [Gantz and Reinsel, 2011]. That is a 100,000% increase in just over 10 years. This rate of increase is expected to continue in the next decade.

1.1 Information Overload and Search Ambiguity

The unprecedented growth of digital resources across the Web has resulted in *information overload*. The overwhelming amount of information has become increasingly unmanageable and created a “poverty of attention and a need to allocate that attention efficiently” [Simon, 1971]. Information overload presents new challenges that continue to draw increasing attention from the information retrieval and data mining communities hoping to innovate new techniques to tackle these new challenges [Dou et al., 2007; Jeh and Widom, 2003a; Ma et al., 2007; Micarelli et al., 2007; Pretschner and Gauch, 1999; Radlinski and Dumais, 2006; Shen et al., 2005; Sieg et al., 2007; Speretta and Gauch, 2005].

Web searches are constantly challenged by the information overload problem.

¹1 exabyte = 10^{18} bytes

²1 zettabyte = 10^{21} bytes

Query terms are inherently ambiguous due to polysemy and most queries are short [Jansen et al., 2000]. Thus, queries are prone to ambiguity of user intent or information needs, resulting in retrieval of many irrelevant documents. As the Web grows exponentially, ambiguity becomes very common and users are in greater need of effective means of disambiguation.

The ambiguity problem can be observed when searching for individuals on the Web. For example, there are 17 entries in Wikipedia for different renown individuals under the same name of “Jim Gray”, including a computer scientist, a sportscaster, a zoologist, a politician, a film director, a cricketer, and so on [Anastasiu et al., 2013]. Similarly, a search query for “Michael Jordan” may refer to the legendary basketball player with six NBA championship titles. It may also refer to a very famous computer scientist. When issuing a query in Google for this famous name in computer science, the first relevant document is his profile page at UC Berkeley which is found in page three ranked 23rd in overall search results.

1.2 Search Personalization

Search personalization is one of the effective ways to address the search ambiguity problem. *Search personalization* allows tailoring and fine-tuning of search results based on individual preferences. Search personalization usually involves building a profile for users resulting in a more accurate search result based on the user’s preferences. Search personalization has become one of the hot topics in the information retrieval (IR) research community [Dou et al., 2007; Jeh and Widom,

2003a; Ma et al., 2007].

Most researches on search personalization make use of techniques that automatically build search profiles from a user search history [Jiang et al., 2011; Liu et al., 2002; Radlinski and Dumais, 2006]. Currently, major Web search engines such as Google³ and Yahoo!⁴ also provide personalized search services. Search personalization techniques used by these major Web search engines typically work in a black box by building user profiles from user search history [Jiang et al., 2011; Liu et al., 2002; Radlinski and Dumais, 2006]. This lack of user control and flexibility can often be inconvenient and counter-productive. For example, such profiles usually capture long-term preferences and would fail to produce good results for short-term information needs [Bennett et al., 2012b; Zamir and Etzioni, 1998]. In addition, multiple categories of preferences, if used together, would compete and work against one another for a particular query, given that queries are inherently and increasingly ambiguous.

1.3 The SPEAR Approach

We introduce a novel approach in search personalization that is *transparent* and allows full user control and manipulation. The *Search personalization with editable profiles* (SPEAR) framework personalizes search results by using one of many user based profiles maintained by users. A *profile* in SPEAR is a logical category that represents some specific area of interest, usually represented by a set of terms. The

³<https://history.google.com>

⁴<http://myWeb.yahoo.com>

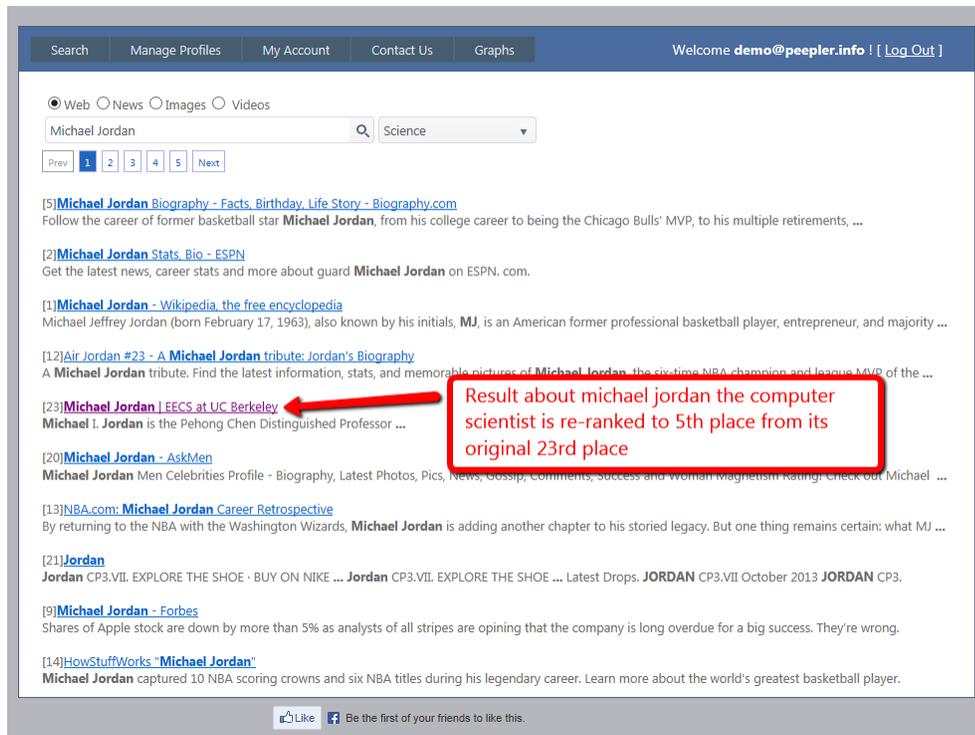


Figure 1.1: A sample search in SPEAR

terms in a profile are phrases that specify user interests. Some examples of profile terms are “car”, “basketball”, “movie theater” and so on. SPEAR re-ranks results retrieved from a search provider by comparing the result documents with the user’s profile.

Figure 1.1 shows a screenshot of a search instance in which a user issues a query for “Michael Jordan” using a science profile and SPEAR re-ranks results about “Michael Jordan” the computer scientist higher than its original rank.

There are various challenges when implementing a *transparent* search personalization approach.

The first challenge is how to build the search personalization profiles. It will

be very difficult for a user to list all preferences and interests. To solve this challenge, SPEAR uses multiple alternative ways to collect user interests to reduce the burden on a user on listing each and every interest. In addition to building profiles from *direct user input*, SPEAR also provides a way to automatically build profiles from *search history* and from user's *social network feeds*.

Furthermore, a specific interest can be expressed in many different ways with different terms. An example would be an interest in “cars”. This interest can be expressed with terms like “cars”, “auto”, “automobile”, “track”, “Vehicle” and so on. When building profiles, the system needs to handle different terms that express the same interest. To solve this problem, SPEAR expands terms to include *semantically similar derivatives*. The automatically generated semantically similar terms, have proven to comprehensively capture user preferences with less input from the user.

Another challenge is how to make profiles easily manageable by users. When giving users full control of profiles, managing large number of profiles and profile terms might be overwhelming. SPEAR provides robust *filtering* and *sorting* user interface that makes managing large set of data manageable. SPEAR also provides two levels of profile management; *basic* profile management and *advanced* profile management. The basic profile management provides an interface for basic profile management tasks like adding and removing profiles, whereas the advanced profile management provides advanced features like controlling how semantically similar terms are generated and managing stop-words. The basic profile management makes the application attractive for a larger audience while the advanced profile

management provides advanced features for more savvy users.

After a profile is built, the next challenge is how to make use of the profile. The basic approach will be trying to compare the search profile with search result documents. Coming up with a comparison algorithm that will use the profile and produce better results is a challenge. If we just use an approach where the algorithm favors only results that contain profile terms, it might have undesirable effects where non-relevant documents that just contain profile terms are ranked higher.

To solve this problem, SPEAR implements an algorithm that builds an inverted index from the profile terms and performs a *cosine similarity* with the result set following the *vector space model* approach. The algorithm also takes into account the original *global ranking* returned by the search provider when producing the new ranking.

SPEAR breaks the black box approach widely used in search personalization techniques and exposes the personalization engine to users. By following a transparent approach in search personalization and giving users full control of how profiles are built, SPEAR has introduced new important features in search personalization.

In SPEAR, users can quickly build a functional profile by adding weighted terms. This is helpful in capturing user's *short-term interest* [Bennett et al., 2012a; Zamir and Etzioni, 1998] where the traditional technique fails as it would take time for such a system to learn user preferences. SPEAR also solves the lack of flexibility exhibited in the commonly used search personalization techniques by providing a

framework where users can create and maintain multiple profiles. Users get to select an appropriate profile when performing a search. Users can also specify a default profile, minimizing the level of effort required to use the system.

In SPEAR, users can specify *negative preferences* by assigning negative weight to terms. Term weights range from -10 to 10, where 10 being most favored and -10 least favored. Search results containing the most positively weighted terms will be promoted whereas search results with negative terms will be penalized and are ranked lower.

SPEAR uses three methods to collect data and build users search profiles resulting in more comprehensive profiles that capture the users' preferences better.

1. *Direct user input* - Users in SPEAR can enter individual profile terms to the system using the profile management interface. When adding individual profile terms, users get to assign the associated weight for the term. The weight measures the importance of a given profile term. As it might be impractical for users to add all important terms into a profile, SPEAR helps by expanding terms to include semantically similar derivatives using a lexical database.
2. *From social network feeds* - In the last decade, social networks have evolved to become an integral part of the Web. Social networks represent logical groups of people that share similar characteristics like background, profession, physical location, interest and more. This logical grouping of people with similar characteristics can be harvested to provide better personalized results

for users. Hypothetically, results that are relevant for a member in the group will most likely be relevant for others in the group as members share similarities which can translate to similar search result preferences. SPEAR has a feature to retrieve feeds from a social network to build out search profiles.

3. *From a user search history* - SPEAR also supports the traditional way of building profiles automatically from user's search history. SPEAR also gives users control over the automatically generated profile.

In summary, SPEAR introduces a novel approach for search personalization that is transparent and is mainly based on editable profiles that are controlled by users. To demonstrate and validate this approach, SPEAR implements all techniques and algorithms discussed as a Web search application utilizing Google search provider and Facebook graph API as a social network feed provider.

CHAPTER 2

BACKGROUND

2.1 Information Retrieval

Information retrieval (IR) is defined as finding material (usually documents) of an *unstructured* nature (usually text) that satisfies an information need from within large collections (usually stored on computers). The keyword “Unstructured”, in this definition, describes the nature of text documents ranging from Web pages to user files involved in information retrieval [Manning et al., 2008]. This distinguishes information retrieval from accessing data in structured relational databases.

Information retrieval may apply data mining concepts like clustering and classification to structure documents or document results based on the user’s information need.

Information retrieval systems can be categorized based on the document collection size to be searched. *Web Search* deals with documents available in the ever-growing Web. For Web search, scalability is an important aspect where the system should be able to serve millions of users searching from billions of documents available on the Web. Web search also needs to consider the structure of HTML documents to link and evaluate relevance of documents. *Personal Search* provides instant search for users searching their own personal data stored locally as document files or remotely as email and cloud data. There are information retrieval systems that provide search results from a specific domain or enterprise network.

This kind of information retrieval systems are similar to Web search but the document collection size to be searched are fewer compared to a Web search.

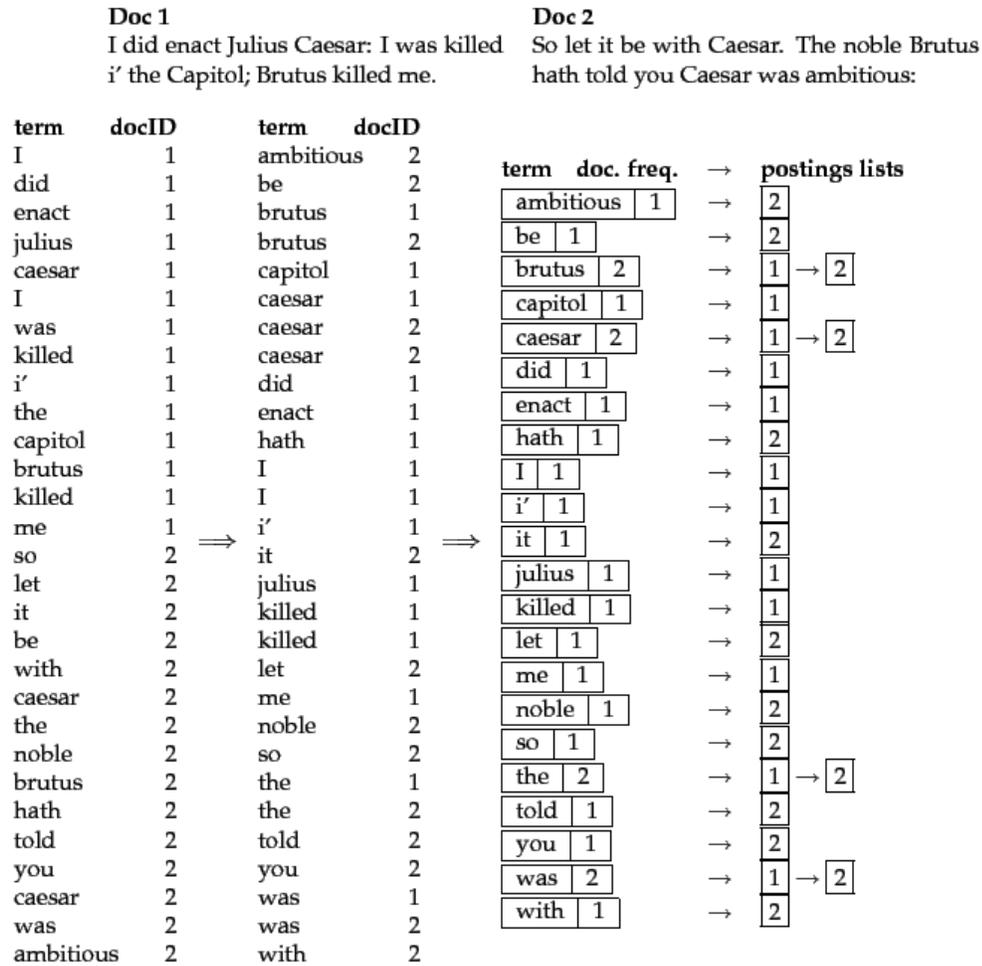


Figure 2.1: Inverted index construction¹

At the heart of an information retrieval system is an *inverted index*, the key data structure supporting efficient query processing. An inverted index consists of a dictionary, alphabetically sorted terms, and postings, a collection of postings list. Each term in a dictionary will point to a posting list. A posting list contains all

¹image courtesy of <http://nlp.stanford.edu>

documents that contain a term in the dictionary. Figure 2.1 illustrates how to build an inverted index. Building an inverted index can be divided into four steps.

1. Identifying the documents to be indexed.
2. Tokenizing the documents by creating a list of terms from each document.
3. Linguistic preprocessing on the list to reduce the list keeping only important terms. Tokenization, removing stop-words, normalization, stemming and lemmatization are some of the main techniques used to produce a list of normalized tokens.
4. Generate the inverted index by creating the dictionary and the posting.

Detailed information on how to build inverted indexes for an information retrieval model can be found in Goker and Davies [2009] and Manning et al. [2008].

2.2 Vector Space Model

An information retrieval model specifies details of *document representation*, *query representation* and *retrieval function* of an information retrieval system. *Boolean model* and *vector space model* are the two popular retrieval models. A Boolean information retrieval model is based on a set theory where all documents containing query terms are returned. Boolean information retrieval systems lack ranking of query results. Any document that satisfies the Boolean query will be returned in a non-ordered format. This is not ideal, especially when searching large number of documents as is the case in a Web search. Users may have to go through thousands

of documents to find relevant documents as results are not ranked based on relevance. A Boolean information retrieval model is an *exact-match* model where only documents that satisfy the Boolean query are returned.

The Vector space retrieval model addresses the main problems mentioned as limitations of the Boolean information retrieval model. This includes:

- Being able to process a *free text query* instead of just returning exact-match documents.
- Being able to rank the search results instead of just returning matching documents.

The document representation in a vector space model supports saving the frequency of terms in each document. This change in document representation introduces ranking to the information retrieval model by promoting documents containing higher frequency of query terms.

Some of the most important concepts in a vector space retrieval model include:

- A *term frequency* denoted by $tf_{t,d}$ is the number of occurrence of a term t in document d
- A *document frequency* denoted by df_t is the number of documents in the collection that contains the term t
- An *inverse document frequency* denoted by idf_t is given by the formula

$$idf_t = \log \frac{|N|}{df_t}, \quad \text{where } |N| \text{ is the document collection size}$$

The term frequency is clearly important for weighting a document against a specific term. A document with high term frequency will be more relevant than a document with lower term frequency for a given query. On the other hand, a document frequency has an inverse relation to the weight of a document with respect to the term. A document that contains a less frequent query term will be more relevant than a document containing a more frequent common query term. For example if we consider the search query “year BMW” in context of a Web search, we want to rank documents that contain the term “BMW” more relevant than documents containing the term “year”, as there will be a lot of documents containing the term “year” making “BMW” the more selective keyword in the query.

The inverse document frequency quantifies this inverse relationship between the document frequency and the term weight. The log function is used to dampen the inverse effect of the document frequency on the weight. By combining the two weighting schemas, $tf_{t,d}$ and idf_t , we can introduce a comprehensive weighting scheme $tf-idf_{t,d}$ given by the formula

$$tf - idf_{t,d} = tf_{t,d} \times idf_t$$

The $tf-idf_{t,d}$ represents the weight of the term t in the document d . With a weighting schema in place, we can represent a document as a vector where the dictionary terms representing each dimension.

$$\vec{V}(d_j) = w_{1,j}\vec{t}_1 + w_{2,j}\vec{t}_2 + w_{3,j}\vec{t}_3 + \dots + w_{i,j}\vec{t}_i$$

where $w_{i,j}$ is the weight of the term t_i in document d_j .

This weight is usually calculated using the tf-idf scheme. This way of representing documents as vectors of term weights is referred to in information retrieval as *vector space model*. Similarity between two vectors can be calculated by measuring the angle between them. If we have two vectors, \vec{a} and \vec{b} , the dot product between the two vectors is given by the formula

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

In the above equation, θ represents the angle between the two vectors. The two vectors are said to be similar when the angle between them is small. In the case when the two vectors are identical, the angle between them is 0 and hence $\cos \theta$ will be 1.

From this observation; we can see that $\cos \theta$ measures the similarity between the two vectors.

$$\text{similarity}(\vec{a}, \vec{b}) = \cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

We can think of a query as a *bag of words* represented in a vector form.

$$\vec{V}(q) = w_1 \vec{t}_1 + w_2 \vec{t}_2 + w_3 \vec{t}_3 + \dots + w_i \vec{t}_i$$

where w_i is the weight of a term t_i in the query. The weight of a term in a query is usually calculated taking the idf of the term in the document collection. After the query is represented as a vector, cosine similarity between the query vector and each document in the collection can be calculated.

$$\text{similarity}(\vec{q}, \vec{d}_j) = \frac{\vec{q} \cdot \vec{d}_j}{\|\vec{q}\| \|\vec{d}_j\|}$$

We can then rank search results based on their cosine similarity value. Documents with high cosine similarity when compared with the query vector will be ranked at the top.

2.3 Information Retrieval System Evaluation

To evaluate the effectiveness of an information retrieval system, there are methodologies developed to measure the effectiveness quantitatively. Information retrieval system evaluation usually utilizes the *relevant* or *irrelevant* property of a document in a search result set. A document is said to be relevant in the search result set if it satisfies the user's information need. In unranked systems, an information retrieval system evaluation model will classify information retrieval systems that return more relevant documents as more effective. With ranked information retrieval systems, we also need to consider where in the search results the relevant documents are ranked.

There are two key attributes used in information retrieval system evaluation that are used to measure the effectiveness of the system. *Precision* of an information retrieval system quantifies what fraction of the returned results is relevant to the information need.

$$Precision = \frac{\text{Number of relevant items retrieved}}{\text{Number of retrieved items}}$$

Recall on the other hand measures what fraction of the relevant documents in the

collection is returned by the system.

$$\text{Recall} = \frac{\text{Number of relevant items retrieved}}{\text{Number of relevant items}}$$

Ideally we want an information retrieval system with a very high precision and recall. In reality, these two attributes of an information retrieval system are inversely proportional.

An information retrieval system that returns all documents in the collection will have the highest recall. Whereas an information retrieval system that does not return any result will have the highest precision value. Clearly precision or recall alone does not give a good measure for an information retrieval. Hence, precision and recall need to be used in combination when measuring an information retrieval system.

F measure is a single measure that uses the weighted harmonic mean of a precision P and recall R .

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}, \text{ where } \beta^2 = \frac{1 - \alpha}{\alpha}$$

The value of β in the F measure indicates whether precision or recall is given more emphasis. When $\beta > 1$ then it indicates that more emphasis is given to the recall. While $\beta < 1$ in the F measure formula indicates emphasis being given to precision. In the special case where equal emphasis is given for both precision and recall, β will have a value of 1 and the F measure formula can be reduced to

$$F_{\beta=1} = \frac{2PR}{P + R}$$

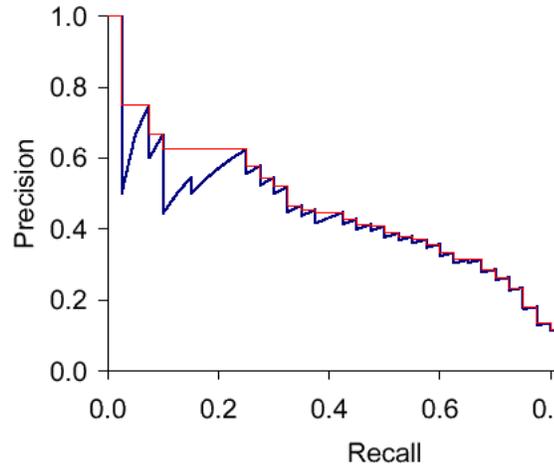


Figure 2.2: Precision/recall graph ²

Precision, recall and the F measure are used to measure the quality of an unranked system. In ranked systems where usually results are returned in a top k fashion, precision and recall are calculated for each set. The *precision-recall curve* is a graphical representation of the precision and recall values collected from each set. As seen from Figure 2.2, the precision-recall curve contains sets of points with the same recall followed by a sharp increase in precision. This is due to the fact that when adding one more irrelevant document to a result set, the precision suffers whereas the recall remains unchanged. The *interpolated precision* is given as

$$p_{interp}(r) = \max_{r' \geq r} p(r')$$

simplifies the precision-recall curve graph where for a given recall, only the highest precision value for all recall values greater or equal to the give recall value is considered.

The Interpolated precision measures precision at all recall levels. For a lot of

²image courtesy of <http://nlp.stanford.edu>

information retrieval applications, especially for Web searches, this might not be a very important measure as what matters to users is how many good results appear on the first few pages. This quality of information retrieval systems is referred as *precision at k*. Precision at k measures precision at fixed low level of retrieved results. The levels used usually range from 10 to 100. Precision at k measure has an advantage on large data sets as it doesn't require knowledge of the size of all relevant documents.

Similar to precision at k , there is a similar measure for recall that measures the recall values at fixed low level of retrieved results. This measure is known as *recall at k*. This is a good measure for Web search applications as it shows the percentage or retrieved documents on the first couple of pages which is important to users.

Another common measure used to measure ranked retrieval systems is the *Mean Average Precision* (MAP). MAP is calculated by averaging the precision over a number of queries. The average precision in each query is calculated by summing up precisions at each relevant document retrieved and dividing it by the number of relevant documents retrieved. When there are no relevant documents retrieved, the precision assigned a value 0. Since MAP only considers precision points where a new relevant document is retrieved, it is sometimes called "average precision at seen relevant documents".

$$MAP = \frac{1}{N} \sum_{j=1}^N \frac{1}{Q_j} \sum_{i=1}^{Q_j} P(doc_i), \text{ where}$$

Q_j is the number of relevant documents for query j

N is the number of queries over which the average precision is going to be calculated

$P(doc_i)$ is the precision at the i^{th} relevant document

2.4 Semantic Similarity

Semantic similarity between terms measures the likeness of the meaning between terms [Fellbaum, 1998; Miller, 1995]. Synonyms represent a group of words that have similar meaning or according to the semantic similarity definition, words with high semantic similarity value. Since a typical language has more than one way of expressing intent, understanding and applying semantic similarity analysis is an important aspect of an information retrieval system. For example, when a user searches for “car”, a good information retrieval system needs to understand the intent of the query and include results for “automobile” automatically.

In SPEAR, search profile terms are automatically expanded to include top semantically similar terms in the search profile. This will help build a better profile with little direct input from the user by expanding the search personalization profile to include concepts similar to what is directly entered by the user.

*WordNet*³, which is a large lexical database of English language, groups words into sets of synonyms which are interlinked to form a semantic similarity tree. In SPEAR, we consider synonyms and “synonyms-of-synonyms” of a word. Based on user preference, either only synonyms or synonyms and “synonyms-of-synonyms” are used to expand user profile terms.

³<http://wordnet.princeton.edu>

CHAPTER 3

LITERATURE SURVEY

With the exponential growth in digital data generated by the world, the need for search engines to dig through this huge amount of data to provide relevant documents is very apparent. As a result, Web search engines have become an integral part of the Web. Despite their popularity, Web search engines commonly provide a “one size fits all” solution with a global ranking algorithm [Allan et al., 2003]. For example, the Google search engine uses *page rank* algorithm [Page et al., 1999] that exploits the linkage structure of the Web to compute global “importance” score. There have been efforts to extend the page rank algorithm to create a personalized view of the Web, redefining importance according to user preference. But it is noted that, it is extremely difficult to realize this approach in practice because of the enormous size of the Web and how the page rank algorithm works [Jeh and Widom, 2003b].

Although, using a global ranking algorithm produces good results in general, it lacks or has inadequate support for personalized results specific to individual users preferences [Maeda et al., 2000].

3.1 Search Personalization

Search personalization is a technique used to filter and re-rank search results using personalization data that is collected *implicitly* [Dou et al., 2007; Jeh and Widom, 2003a; Ma et al., 2007; Micarelli et al., 2007; Pretschner and Gauch, 1999] or

explicitly [Noll and Meinel, 2008]. Implicitly collected personalization data includes users search history [Matthijs and Radlinski, 2011], users click through and any personalization data collected in the background. Explicitly collected personalization data includes documents marking with relevant or irrelevant tags and search profiles that are built and maintained explicitly by users. When building personalization profiles implicitly from users search history, there is usually a concern about violating users privacy [Qiu and Cho, 2006; Shen et al., 2005; Sieg et al., 2007; Speretta and Gauch, 2005; Sun et al., 2005]. By giving users the power to remove and add search personalization data, we can achieve a transparent system where users have full control.

There has been a lot of effort from the research community to study and propose innovative ways to provide personalized search results for users [Bao et al., 2007; Jeh and Widom, 2003a]. The challenge in designing a search personalization system is that, it is very difficult to accurately capture users' interest as it is somewhat a moving target. A user's information need for a given query may vary depending on where and when the user issued the query.

For example, a user might issue a search query for "mouse". The search results may contain resources about a computer peripheral "mouse" or it could be about the small mammal "mouse". If we are using a search profile and we know that the user is interested in Biology, we may rank search results about the small mammal "mouse" higher. However, what if this user who is interested in Biology is shopping for a computer at the time when the search query is issued? So when building a

search profile for search personalization, it is very important to capture both *short-term* and *long-term* interests [Bennett et al., 2012a].

Most of the researches on search personalization focus on search personalization techniques that use implicitly collected data [Sugiyama et al., 2004; Teevan et al., 2005]. In fact, popular search engines like Google and Yahoo follow this approach to provide personalized search results for their users. The implicitly collected data is usually users search history. With this approach, the search engine collects users search history by capturing the query submitted and links that are clicked. When a user issues a query, the URLs previously clicked on by the same user for the same query are promoted [Dou et al., 2007; Matthijs and Radlinski, 2011].

Although, this approach has the advantage of not requiring a lot of effort from the user, it has some drawbacks due to the lack of control on profiles. Profiles constructed with this approach usually capture long-term preferences and would fail to produce good results for *short-term information needs* [Bennett et al., 2012a; Zamir and Etzioni, 1998]. Moreover, the profiles built with these techniques fail to systematically or manually capture users' *negative preferences* which can be very useful in filtering out undesirable results. These techniques also lack *flexibility* to accommodate users' interest changes as they provide only one search profile per user.

There have been more researches on search personalization approaches that use implicitly collected data than researches involving search personalization approaches that use explicitly collected data [Noll and Meinel, 2008]. Noll and Meinel's

approach to Web search personalization consists of social bookmarking and tagging. In this approach, a user will explicitly bookmark important links. A user also gets to tag these resources with keywords. The bookmarks and tags are then used to re-rank search results. Although, this approach has user-controlled bookmarks and tags, it does not support negative preferences. It also does not have a way to automatically expand keywords. SPEAR addresses these two concerns by means of negative weights for terms and automatically adding semantically related terms.

3.2 Social Search

Millions of users around the world have integrated social networking sites into their daily routines [Backstrom et al., 2006]. Social networks represent connections among people that share similarities. Individual and group behaviors can be mined from social networks. This provides a great opportunity for many researchers in various fields.

A social graph represents a social network as a graph. In which, the vertices represent the users and the edges on a social graph represent the relationships among users [Ugander et al., 2011]. Social search uses the power of social networks to provide a personalized search result to users [Bao et al., 2007; Carmel et al., 2009; Noll and Meinel, 2008].

The collaboration in social search can be explicit or implicit. Users can explicitly annotate results and the annotations are used to re-rank results [Bao et al., 2007]. Search results clicked by users are collected and are used to personalize

search results of other users in the social circle. A profile can also be constructed from users' social network feeds. Users' social network feeds include resources shared by other users in the social graph or direct posts made by the user.

The authoritative power of a specific resource depends on how popular the resource is in the social network. One of the challenges in search personalization using user's profile is how to handle a query request issued for the first time outside the user's profile domain knowledge. This problem can be solved by using social networks to build search personalization profiles that derive unlearned preferences from other users.

CHAPTER 4

THE SPEAR APPROACH

The SPEAR approach for search personalization requires designing and implementing a Web application with a back-end database. The framework utilizes Google search API and Facebook graph API. Google and Facebook are selected for their popularity. The implementation is independent of the search and social network feed providers. SPEAR can be easily adapted to other providers. The quality of SPEAR is measured using standard information retrieval system evaluation measures discussed in section 2.3 (Information Retrieval System Evaluation). This section will discuss the architecture of SPEAR and its modules.

4.1 SPEAR Architecture

The SPEAR design consists of five functional modules, the *profile management module*, the *query processing module*, the *search history module*, the *social network feed module* and the *personalization module*. When a user issues a search query q to SPEAR, the query processing module makes an API call to the search provider¹ and retrieves multiple paged results. The results returned from the search provider are aggregated and passed to the personalization module. If the user has the “Extract Terms from Search History” setting enabled, the search query q is passed to the search history module. After applying stop-word removal, the search history module generates a set of terms t and passes it to the profile management module. If a

¹Google search provider is used for demonstration and it returns a maximum of 64 search results.

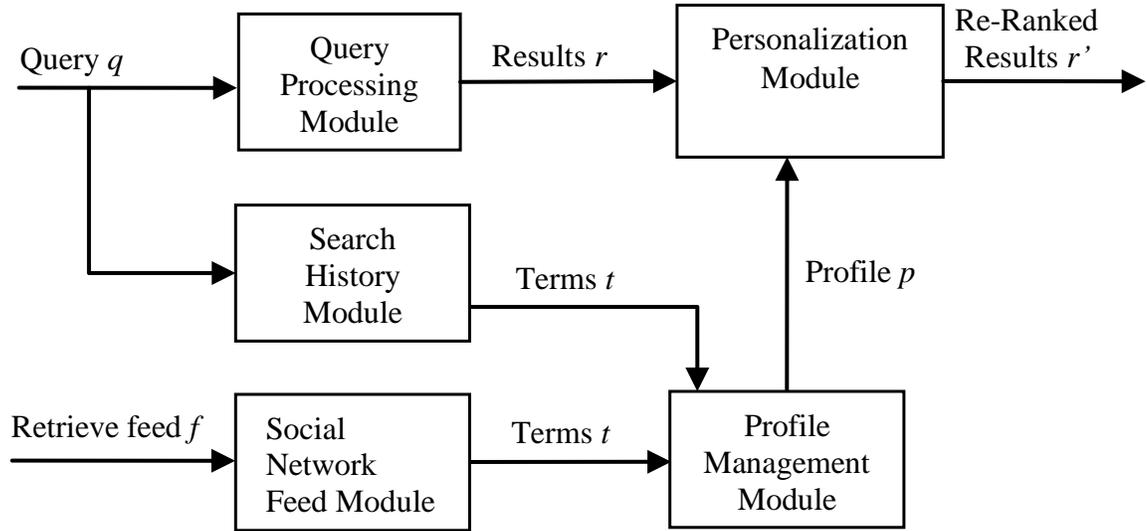


Figure 4.1: SPEAR functional modules

profile p is selected when issuing the search query, the profile management module constructs an inverted index from the selected profile using term weights.

In SPEAR, profiles are constructed from three sources. If the social network integration setting is enabled for the user, the social network feed module will retrieve live feeds from the social network linked to the user account. The social network feed module then cleans the data by applying stop-word removal and constructs an inverted index from the feeds. Then the top ranked terms t are passed to the profile management module. Users also contribute to the profile construction directly by adding, modifying and deleting profile data. The profile management module provides the user interface and functionalities needed to manipulate and manage profiles.

The personalization module takes the search results r and the profile data p

generated by the profile management module to produce a new ranking. Search results are re-ranked and a new search results r' are generated by doing a cosine similarity analysis between the search results r and profile data p .

4.1.1 Profile Management Module

The profile management module provides a way to create and manipulate profiles. A profile is a logical category that represents some specific area of interests. An example can be a “Science” profile. A user in SPEAR can have multiple profiles and can select a specific profile when performing a search. A profile contains terms, which are specific topics that a user likes or dislikes. A user can choose an initial weight for a term, which can range from -10 to 10. Documents containing terms with higher weight are ranked higher. A negative weight for terms indicates that the user would prefer to rank search results that contain the term lower.

When creating a profile, the user will get to choose to either extract profile terms from a social network or from a search history. When adding terms to a profile, SPEAR automatically adds semantically related terms with a distance threshold of 1 to the profile. The weight of these automatically added terms depend on their distance from the main term in a semantic similarity tree. i.e. if a term t has a weight w , then the weight of a semantically related term t' at distance of h from t in the semantic similarity tree will have a weight of w' , where w' is given by the formula

$$w' = \frac{w}{h}$$

In SPEAR, profiles are constructed from three sources.

1. *Direct users input* - a user in SPEAR can enter individual profile terms to the system using the profile management interface. When adding individual profile terms, the user gets to assign the associated weight for the term. The weight measures the importance of a given profile term. In the advanced profile management section, the user can configure how the application automatically adds semantically similar terms.
2. *Social network feeds* - SPEAR has a feature to retrieve feeds from a social network to build profiles. If a user has the social network integration setting enabled, the SPEAR system automatically gets feeds from the user's social network and updates the profile. For demonstration purpose, SPEAR supports integration with Facebook social network. The advanced profile management section in SPEAR lets a user define a profile level set of stop-words. When analyzing a feeds from a social network, SPEAR applies the set of stop-words and constructs an inverted index from the feeds using the *lucene indexing framework*. The weight of a term t in a social network post p is calculated based on the weighting formula

$$w_{tp} = tf_{tp} \times \log \left(\frac{\#likes_p + \#comments_p}{d_p} \right), \text{ where}$$

w_{tp} is the weight of the term

tf_{tp} is the term frequency of the term in the post

$\#likes$ is the number of likes of the post

$\#comments$ is the number of comments of the post

d_p is the distance of the owner of the post from the user in the social graph

The term weighting formula used tries to capture the most frequent terms in popular posts from close friends. Popularity in a social network post can be determined by the number of “likes” and “comments” the post receives. The log function is used to dampen the effect of the number of “likes” and “comments” on the term weight. After the weight for each term has been calculated, then most important terms with higher weights are added to the profile.

3. *Users search history* - SPEAR also supports a way to build profiles automatically from a user’s search history when the setting to extract search history is enabled. When extracting profile terms from users search history, an inverted index is constructed from the search history after removing stop-words. The frequency of the term in the inverted index is taken as the weight of the term in the profile.

The profile management module combines the profile terms from the three sources to generate a search profile. A vector is constructed from terms in the profile along with their weight. Then, the constructed vector is passed to the personalization module to re-rank search results.

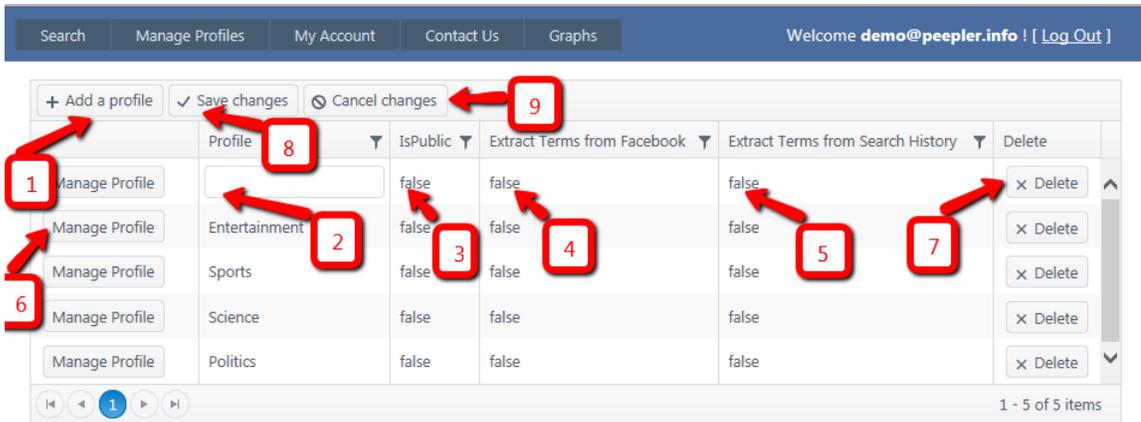


Figure 4.2: Profile page anatomy

1. Add profile - This presents users with an interface to add a new search profile.
2. Profile name - Profiles have unique names that will identify them. The names are entered when creating a new profile and can also be changed later.
3. Profile privacy setting - This setting allows users to control the privacy of their profiles. When profiles are marked as private, they are not shared by other users.
4. Social network integration - When Facebook integration is enabled, profile terms are extracted from the social network feeds.
5. Search history tracking - When users enable this setting, SPEAR will analyze users' search history and uses the information extracted to build search profiles.
6. Manage profile link - This takes users to a page where they can add, edit or delete profile terms.

7. Delete profile - Users can delete profiles that they don't need any more.
Deleting a profile removes the associated terms and stop-words.
8. Save changes - All user interfaces in the profile management module implement transactional changes. Save changes commits changes made to profiles.
9. Cancel changes - All user interfaces in the profile management module implement transactional changes. Cancel changes rolls back changes made to profiles.

The Profile Management Module consists of three main sections.

1. *Profiles Management Section* - In this section, SPEAR provides a user interface for adding, deleting and modifying profiles. SPEAR maintains a transaction for user changes applied to profiles. Changes will be committed only when the "Save changes" command is issued. Changes can be rolled back by issuing a "Cancel changes" command.
2. *Profile Detail Section* - The profile detail section provides a user interface for managing profile terms. A user can add, delete, modify or mark terms as stop-words.
3. *Advanced Profile Settings Section* - The advanced profile settings section provides a user interface for controlling the advanced features in SPEAR. The advanced features include controlling the distance threshold for semantically generated terms and managing stop-words.

4.1.2 Social Network Feed Module

The *social network feed module* is responsible for extracting feeds from users' social network. In order for SPEAR to be able to extract feeds, users need to link SPEAR with their Facebook account. This can be done by either logging in to SPEAR with a Facebook account or by adding Facebook account information in SPEAR's user account information section. SPEAR periodically scans users' social network to retrieve new feeds. New feeds are retrieved using the Facebook graph API². SPEAR analyzes the feeds and extracts important terms. Stop-words are used to remove common terms and an inverted index is built from the feeds. SPEAR creates an inverted index using *lucene*³ by assigning weights according to the formula.

$$w_{tp} = tf_{tp} \times \log \left(\frac{\#likes_p + \#comments_p}{d_p} \right), \text{ where}$$

w_{tp} is the weight of the term

tf_{tp} is the term frequency of the term in the post

$\#likes$ is the number of likes of the post

$\#comments$ is the number of comments of the post

d_p is the distance of the owner of the post from the user in the social graph

The formula favors resources that are popular in the user's social network. Once the terms are ranked according to their weight, the highest ranked terms are passed to the profile management module to be added to the user profile.

²<https://developers.facebook.com/docs/graph-api>

³Lucene is an open-source project for indexing and searching documents. More information about lucene is found at <http://lucene.apache.org/>

4.1.3 Query Processing Module

The query processing module takes a query q and gets Web search results using a search engine (e.g. Google). For better performance, the search API is accessed from the client side using JavaScript instead of executing the search on the server side and returning the result to the browser. If no profile is used, then the result returned from the search engine is directly displayed to the user. On the other hand, if a profile is used, then the personalization module takes the search results along with the profile data and returns the re-ranked results.

4.1.4 Search History Module

The search history module takes a query q to generate profile terms. After removing stop-words, the search history module normalize, stem and lemmatize query terms using the lucene indexing framework to generate a set of terms t . The terms generated are then passed to the profile management module to be added to the user profile.

4.1.5 Personalization Module

The personalization module uses a tf-idf weighting scheme. A vector space model is constructed from the search results by calculating the frequency of each term in the selected profile within each document in the search results. Each search result's *snippet* and *title* forms the document content for each result. For the profile, the weight of each term is used as the tf-idf weight of that term in the profile document.

The personalization module uses cosine similarity to find the similarity between the profile and the search results. The results are re-ranked based on the similarity score evaluated. The original document rank returned from the search provider is considered by SPEAR when re-ranking results.

$$w_{dp} = \text{cosinesimilarity}(d, p) \times \log \frac{|N|}{d_r}$$

w_{dp} is the new weight of a document d with respect to a profile p

p is the profile used

d_r is the rank of the document in the original search results

N is the number of documents returned in the original search results

CHAPTER 5

SPEAR IMPLEMENTATION

SPEAR is implemented in a context of a Web search. A web search application is built to demonstrate the approach. The Web search application will use a *Google* search provider. The search personalization framework uses *Facebook* as a social graph provider. As the re-ranking of the search results based on the user profile needs to happen for each search request, performance is really important. The similarity comparison algorithm is optimized to re-rank the results efficiently. In this section, we will discuss the detailed implementation of SPEAR including programming languages, databases, and frameworks used.

SPEAR is implemented as a client server solution with services using popular Web technologies like *WCF*, *ASP.NET*, *AJAX*, *JQuery*, *JSON* and other similar web technologies. SPEAR uses *JavaScript* for client side scripting. For server side code, SPEAR implemented a Microsoft .NET solution with C# as the preferred choice of programming language. For a local repository, SPEAR uses a database that runs in Microsoft SQL Server 2008 Express Edition. An instance of SPEAR application is installed on a Texas State University Computer Science Department's Web Server. The application is available at <http://dh231b01.cs.txstate.edu/spear>. This Web site is accessible within the Texas State University network.

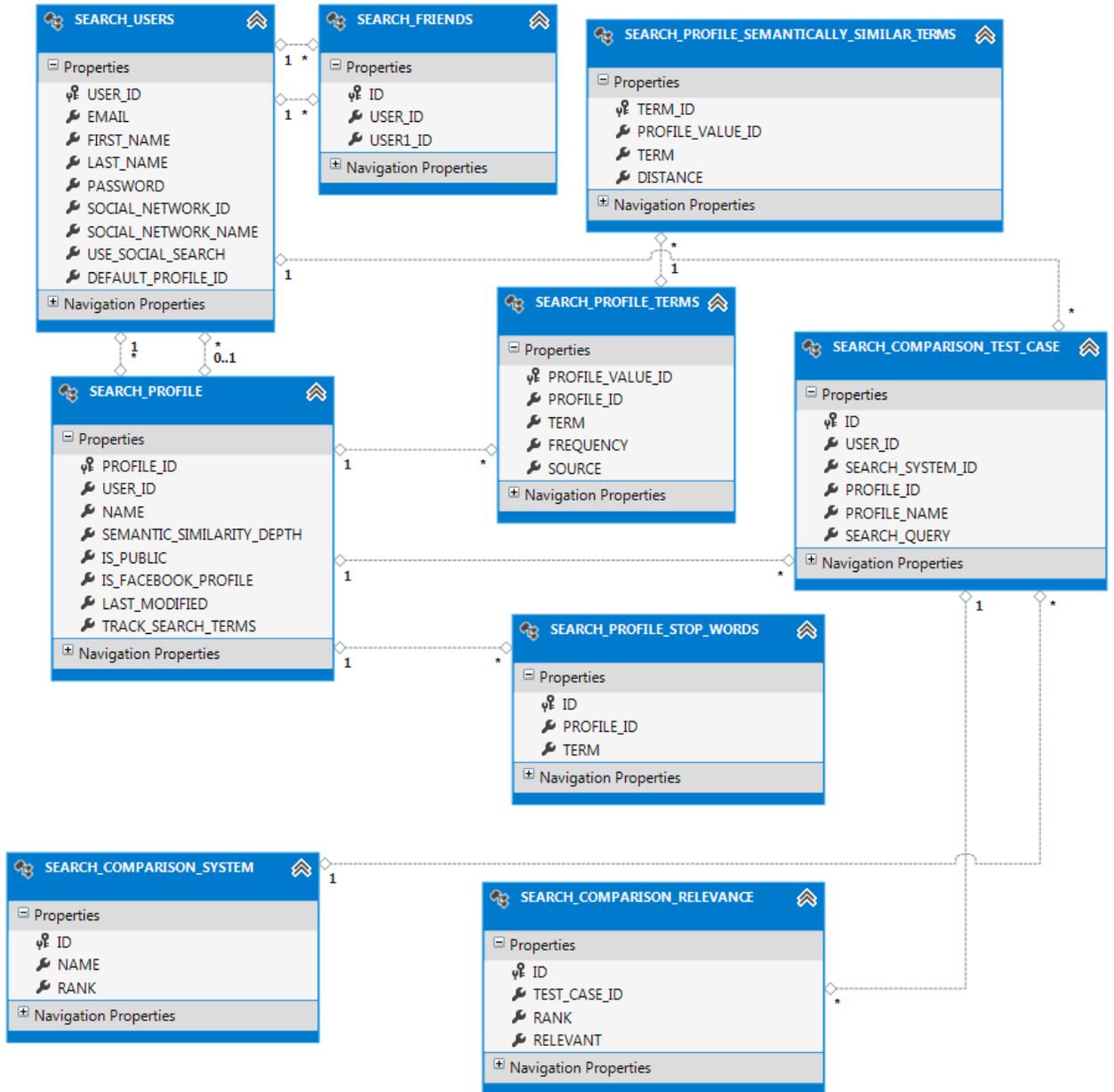


Figure 5.1: SPEAR database entity relationship diagram

5.1 Database Design

The design includes a database system that is used to store users' information and search profiles. The database design also supports a way to store search results and

relevance feedback from users. It contains nine tables which are related via different relationships. The main database entities in the system include:

- SEARCH_USERS - This entity represents SPEAR users. Users can create and maintain one or more search profiles and friends.
- SEARCH_PROFILE - The search profile is the most important entity in the system. It represents the logical unit of users' search interest. A profile groups a list of terms that are important in the search context.
- SEARCH_COMPARISON_SYSTEM - The search comparison system entity represents the various IR systems, including the SPEAR system, used in evaluating the framework. The systems used to compare SPEAR include, the Google search provider and the Google personalized search provider systems.

5.2 Object Relational Mapping

For object relational mapping, SPEAR uses ADO.NET *Entity Framework*. Entity Framework allows mapping relational data using domain specific objects. This approach eliminates the need to write SQL statements for most of the data access layer. Entity Framework also removes the dependency between the domain specific objects and the data access implementation details, making the choice of the database used very flexible.

5.3 Web Services

SPEAR follows a service oriented client server architecture where the application provides Web services for accessing and manipulating data. By removing the Web layer from any data access code and exposing that functionality through Web services, a separation of concern was achieved. This also produced a reusable service that can be consumed from any client applications. There are two main Web services that expose the data layer: Personalization service and CRUD service.

5.3.1 Personalization Service

The personalization service exposes functionalities that are related to SPEAR's personalization functions. This includes getting formatted search profile data, generating semantically similar terms and so on. The service is implemented as a WCF (Windows Communication Foundation) service that returns JSON serialized objects. The WSDL (Web Services Description Language) for the service is found at <http://dh231b01.cs.txstate.edu/SPEAR/SearchProfileService.svc?singleWsdl>. Some of the important methods exposed through the Web service include:

- `List<ProfileTerm> GetSelectedProfileTerms(int profileId, string queryTerm, bool trackSearchQuery)` - This method is called from the search page through a JavaScript proxy class generated to access the Web service. It passes the `profileId` that uniquely identifies the profile used along with the search query terms. Then, the system builds a list of terms along with their calculated weights. The list is then serialized as a JSON object and is returned to the

Web client.

- `string TrackQueryTerm(int profileId, string queryTerm)` - This method, as the name suggests, adds a search query term to a user search profile.
- `void UpdateFaceBookProfile(List<FaceBookFeed> feeds)` - This method accepts a list of feeds retrieved from Facebook for a user. The method analyzes the feeds and extracts important terms.

5.3.2 CRUD Service

The CRUD service exposes create, update and delete functionalities for the SPEAR data entities. This includes adding, updating and deleting profiles, profile terms, stop-words and so on. The service is implemented as a WCF (Windows Communication Foundation) Service that returns JSON serialized objects. The WSDL (Web Services Description Language) for the service is found at <http://dh231b01.cs.txstate.edu/SPEAR/CRUDService.svc?singleWsdl>. Some of the important methods exposed through the Web service include:

- `IEnumerable<SearchProfile> CreateProfile(IEnumerable<SearchProfile> profiles)` - This method creates a new profile.
- `DataSourceResult ReadProfile(int skip, int take, IEnumerable<Sort> sort, Filter filter)` - This method returns a list of available profiles.
- `string UpdateProfile(IEnumerable<SearchProfile> profiles)` - This method updates an existing profile.

- string DestroyProfile(IEnumerable<SearchProfile> profiles) - This method deletes a profile.
- IEnumerable<StopWord> CreateProfileStopWord(IEnumerable<StopWord> profileStopWords) - This method creates a new stop-word.
- DataSourceResult ReadProfileStopWord(int skip, int take, IEnumerable<Sort> sort, Filter filter) - This method returns a list of available stop-words.
- IEnumerable<StopWord> UpdateProfileStopWord(IEnumerable<StopWord> profileStopWords) - This method updates an existing stop-word.
- string DestroyProfileStopWord(IEnumerable<StopWord> profileStopWords) - This method deletes a stop-word.

5.4 Web Pages

The Web pages in SPEAR are built using ASP.NET. The ASPX files mainly define the layout for the Web pages. They have the session validation server side code to track the user's identity. Most of the interactions on the page are handled with a client side JavaScript that will call a WCF service for any server side actions.

CHAPTER 6

EVALUATION

The SPEAR evaluation involved generating test queries and performing standard information retrieval system evaluation methods discussed in section 2.3 (Information Retrieval System Evaluation). For comparison purpose, test queries were run in three systems: Google Non-Personalized, Google Personalized and SPEAR. Results from each system were captured and stored in the SPEAR database for analysis purpose. 11-point interpolated average precision, precision at K , recall at K and MAP (Mean Average Precision) were calculated and analyzed to compare SPEAR with the other systems.

- *Google Non-Personalized* - The Google Non-Personalized system was one of the systems evaluated using the test cases. The Google Non-Personalized system refers to the standard Google search available without logging in to the Google system. It is also the same system that returns search results when using Google search API. It returns search results based on a global ranking algorithm. All users using Google Non-Personalized system will see the same result when issuing same search queries.
- *Google Personalized* - The Google Personalized system was a comparable system to measure SPEAR as it provides personalized search results. The Google Personalized system is available for users with Google account who are logged into Google when performing a search. Google tracks a user's search

history. When a user requests the same search query, Google ranks results that the user has previously visited higher. Although, we do not know the internals of the search personalization algorithm used by Google, its behavior suggests it is favoring links that a user has visited with a previous search. Logged in users in Google can view their search history by going to the search settings page¹. The Google test account used for the Google Personalized search is an active account that was created over 5 years ago. The account had over 9,960 Google searches captured in the Google search history. The search categories in the search history were diverse. Some search queries in the history were informational queries like “Miami”, “Wedding” and so on. Others were navigational queries like “Texas state university”, “Texas DMV” and so on. There were also transactional queries like “download antivirus”, “purchase data mining text book” and so on.

- *SPEAR* - An instance of the Web search personalization application that was built to demonstrate *SPEAR* was installed on the Texas State University Computer Science Department Web server². A demo user account was created with four profiles: Entertainment, Politics, Science and Sports. These profiles were built with direct user inputs in less than five minutes. Each profile contains about six terms on average. A complete list of the profiles used for testing is shown in Table 6.1.

¹<https://history.google.com>

²<http://dh231b01.cs.txstate.edu/spear>

Table 6.1: Test profiles

Profile	Terms
Entertainment	Actor, Artist, Band, Comedian, Grammy, Journalist, Movie, Music, Oscar, Sing, Song
Politics	Congress, Democrat, Election, Minister, Parliament, Politics, Republican
Science	Biology, Computer, Graph, Machine, Physics, Research, Science
Sports	Baseball, Basketball, Fifa, Football, MLB, NBA, NFL, Soccer, Sport

6.1 Test Cases

To collect feedback on the search results, SPEAR’s search interface was updated so that a user can mark relevant documents and submit search results feedback.

Twenty test cases were run on the three different systems. The test case search queries were selected from a pull of ambiguous popular people names³. Ambiguous queries are important in measuring search personalization systems as they represent more than one topic. The personalization module then tries to select the appropriate topic based on the user’s profile.

³<http://www.worldwideinterWeb.com/item/5548-pro-athletes-with-the-same-name-as-celebrities.html>

Table 6.2: Test cases

Search query	Profile used	Information need detail
Dave Clark	Sports	About a baseball player
Dave Clark	Entertainment	About a lead singer of a band
Tito Jackson	Entertainment	About a singer from Jackson five
Tito Jackson	Politics	About a politician
Dave Stewart	Sports	About a baseball player
Dave Stewart	Entertainment	About a singer from the 80's
Kevin McHale	Sports	About a basketball player and coach
Kevin McHale	Entertainment	About an actor
Allan Campbell	Science	About a biologist
Allan Campbell	Sports	About a rowing sports person
Ted Williams	Entertainment	About a homeless voice over artist
Ted Williams	Sports	About a baseball player
Davy Jones	Entertainment	About a musician
Davy Jones	Sports	About a baseball player
Mike Wallace	Entertainment	About a journalist
Mike Wallace	Sports	About an American football player
Adam Scott	Entertainment	About a TV star
Adam Scott	Sports	About a golfer
Michael Jordan	Science	About a person who is famous in machine learning field
Michael Jordan	Sports	About a basketball player

6.2 Results

After test queries were run and results were captured, graphs were plotted to visualize and study the results. When analyzing results, the search results from Google search API are considered as the document collection for the three systems to be evaluated.

6.2.1 11-Point Interpolated Average Precision

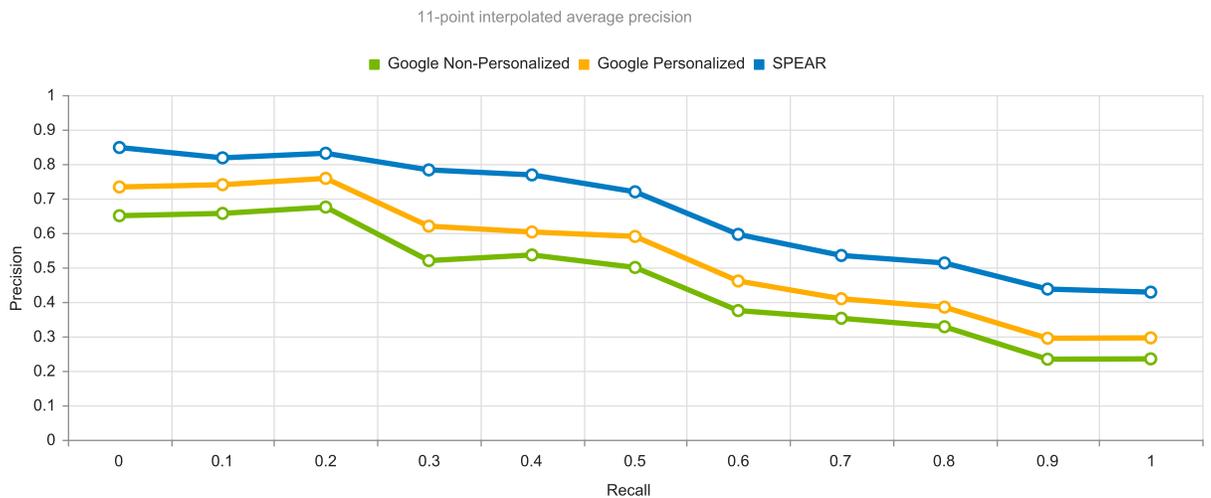


Figure 6.1: 11-point interpolated average precision

Figure 6.1 shows the 11-point interpolated average precision graph for the three systems: Google Non-Personalized, Google Personalized and SPEAR. We can clearly observe that for each recall point in the recall axis, Google Personalized has a higher precision value than Google Non-Personalized. We also see that SPEAR has the highest precision value for each recall point compared to both Google Non-Personalized and Google Personalized systems. When SPEAR re-ranks search

results from Google Non-Personalized system, it moves more relevant results to the top, resulting in a higher precision for each relevant document retrieved.

6.2.2 Precision at K

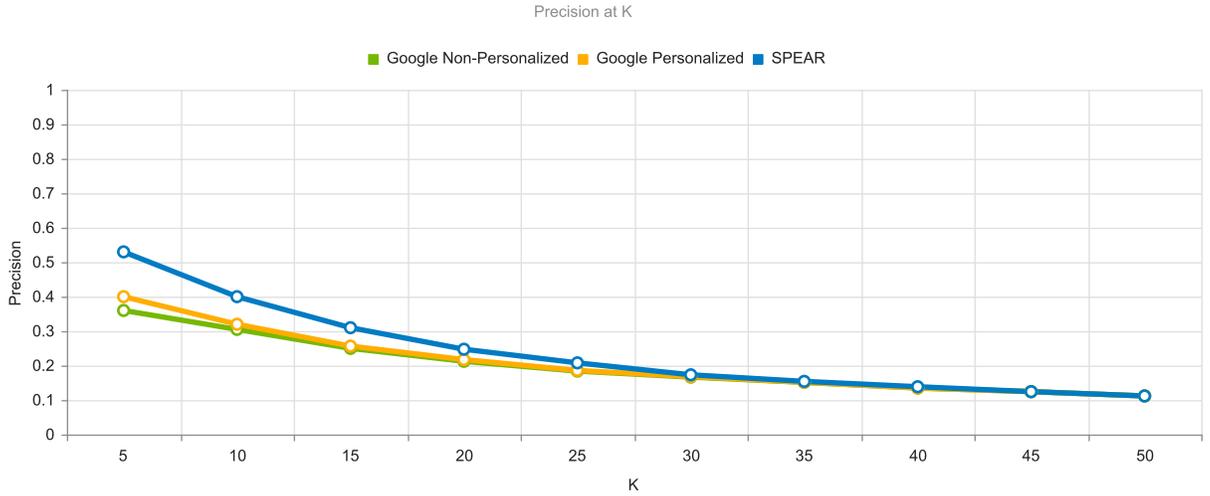


Figure 6.2: Precision at K

Figure 6.2 shows the average precision at each top K items retrieved for the three systems evaluated: Google Non-Personalized, Google Personalized and SPEAR. K ranges from 5 to 50. From the graph, we can see the typical trend observed in a Precision at K graphs where the precision is inversely proportional to the number of documents retrieved. In a typical Web search, relevant documents are usually found mostly in the first couple of pages. Due to this behavior, as more documents are retrieved the proportion of relevant documents to the total documents retrieved decreases.

From the graph, we can also observe that until $K = 30$, SPEAR on average

has a higher precision value compared to both Google Non-Personalized and Google Personalized. Google Personalized has a slightly higher precision than Google Non-Personalized. The result shows that, SPEAR brings more relevant documents at the top compared to Google Non-Personalized. We also see that SPEAR produces better personalized search results compared to Google Personalized.

For $K \geq 30$ in the graph, all three systems start to converge and have more or less similar precision values. This is due to the fact that the personalization framework uses the same set of search results. SPEAR rearranges the search results by bringing more relevant documents at the top and almost all relevant documents are found within the first three pages. For $K \geq 30$, all three systems have equal number of relevant documents as they share the same document collection.

6.2.3 Recall at K

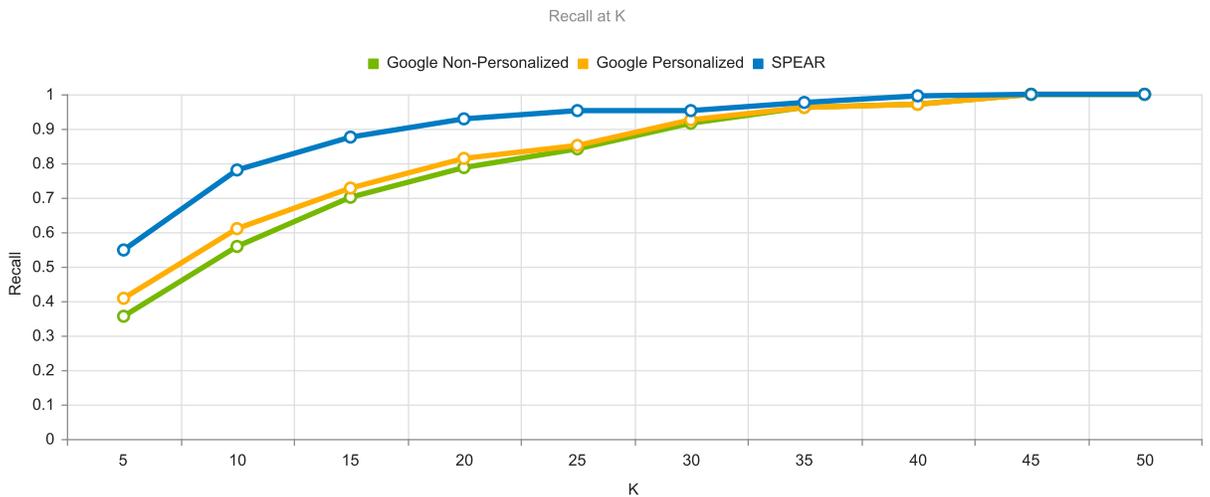


Figure 6.3: Recall at K

Figure 6.3 shows the average recall for the test case queries at each top K items retrieved for the three systems evaluated: Google Non-Personalized, Google Personalized and SPEAR. K ranges from 5 to 50. From the graph, we can see the typical trend observed in a recall at K graphs where the recall is directly proportional to the number of documents retrieved. Since recall is only concerned with the percentage of the relevant documents retrieved, adding more documents usually increases the recall until all relevant documents are retrieved.

From the graph, we can also observe that until $K = 30$, SPEAR on average has a higher recall value compared to both Google Non-Personalized and Google Personalized. Google Personalized has a slightly higher recall than Google Non-Personalized. On the $K \geq 30$ section of the graph, we can see that all three systems start to converge to the maximum recall value of 1 as all relevant documents are more or less retrieved.

6.2.4 MAP (Mean Average Precision)

Table 6.3: MAP values

	MAP
Google Non-Personalized	0.4475
Google Personalized	0.5234
SPEAR	0.6504

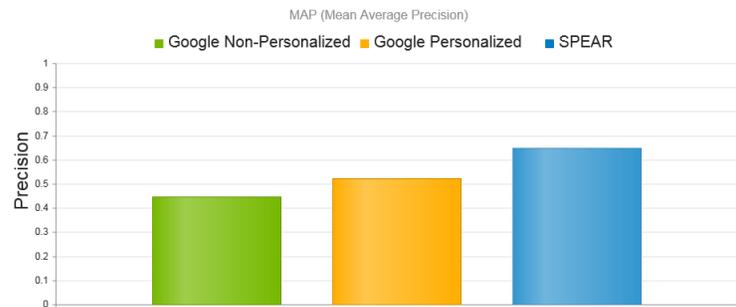


Figure 6.4: MAP

Table 6.3 shows the MAP (Mean Average Precision) for the three systems. We can observe that SPEAR has the highest mean average precision value whereas Google Non-Personalized has the lowest. Google Personalized has slightly higher mean average precision value than Google Non-Personalized.

6.3 Discussion

Test results show that SPEAR produces better personalized search results compared to Google Personalized search results, demonstrating the potential of SPEAR.

Possible improvements to SPEAR include enhancing the profile terms extraction from a social network. The approach used by SPEAR to extract profile terms from a social network involves identifying most frequent terms in popular posts from close friends. This approach is prone to mistakenly label irrelevant terms as relevant just because they are frequent. Although, applying stop-word removal when extracting terms from social network feeds reduces the amount of irrelevant terms extracted, it is not a complete solution. A better approach could be trying to summarize posts to capture important phrases using an approach similar to Sharifi et al. [2010].

CHAPTER 7

CONCLUSION

With wide availability and popularity of the Web, the digital information generated by the world is growing at an exponential rate. With this growth rate, the need to deliver quality information to users has become the main focus of the information retrieval community. There has been a lot of progress in providing quality search results. Most of the work has been focused on implementing a global “one size fits all” ranking algorithm. There has also been significant amount of work done in providing personalized search results for users. These approaches, although producing better results than their corresponding non-personalized versions, work in a black box and are far from being perfect.

We propose SPEAR, a transparent search personalization framework that gives users full control over how profiles are built. SPEAR also combines different sources for users’ profile data generation. Being able to extract users’ preferences from a social network is promising and can be studied in more detail in the future. The SPEAR approach has produced very promising results and could be evaluated and adopted on a larger scale. SPEAR’s ability to support negative preferences is another promising direction that needs to be pursued.

APPENDIX A

SPEAR INSTALLATION AND ADMINISTRATION

This section explains how to install and administer SPEAR. The installation and administration section lists the system requirements and describes how to install SPEAR on a windows operating system.

Table A.1: SPEAR system requirements

Requirements	
Operating System	x86 or x64 Windows 7 (Professional, Enterprise and Ultimate), Any edition of Windows 8, x86 or x64 Windows Server 2008 or above.
Processor Type	x64 Processor: AMD Opteron, AMD Athlon 64, Intel Xeon with Intel EM64T support, Intel Pentium IV with EM64T support x86 Processor: Pentium III-compatible processor or faster
Processor Speed	x86 Processor: 1.0 GHz or faster x64 Processor: 1.4 GHz or faster
Memory	1GB or more
.NET Framework	NET 4.0 or above installed on the system
Database	Any edition of SQL Server 2008 or above.
IIS	IIS 7 or above with WCF installed and configured

SPEAR is implemented to run in a windows environment. The application is developed using Microsoft .NET Framework 4.0 and runs in a Windows Internet

Information Services (IIS) server. The system requirements to install and run SPEAR are listed in Table A.1.

Steps to follow when installing SPEAR

1. Download the source code from
<http://dh231b01.cs.txstate.edu/spear/source.zip>
2. Extract downloaded source code to a local folder. E.g. C:\SpearSouceCode
3. Install the SPEAR database by opening the command prompt and running the command, “sqlcmd -S myServer\master -i C:\SpearSouceCode\spear.sql”.
Where myServer refers to the instance of the SQL server you want to install the SPEAR database to.
4. Go to the folder you extracted the source code to and edit the web.config file.
Update the “PersonalizedSearchEntities” connection string with a user credential that has access to the newly created SPEAR database.
5. Check the database installation making sure the login used in the web.config file has access to the SPEAR database. You can verify the database login by running the command “sqlcmd -U UserLogin -S myServer\spear”
6. Open the command prompt and run msbuild command for the SPEAR solution.
“msbuild C:\SpearSouceCode\PersonalizedWebSearch.sln
/p:Configuration=Release /p:DeployOnBuild=True

```
/p:DeployDefaultTarget=WebPublish /p:WebPublishMethod=FileSystem  
/p>DeleteExistingFiles=True /p:publishUrl=c:\Spear"
```

7. Stop IIS by running the command “net stop WAS”
8. Install the SPEAR web application by running the appcmd add command.
E.g. To install SPEAR under the default web site, run “appcmd add app
/site.name:“Default Web Site” /path:/spear /physicalPath:c:\Spear”
9. Start IIS by running the command “net start W3SVC”
10. Check the installation by going to <http://<yourhostname>/spear>. For
example if SPEAR is installed under the “Default Web Site”, the URL will be
<http://localhost/spear> .

REFERENCES

- Allan, J., Aslam, J., Belkin, N., Buckley, C., Callan, J., Croft, B., Dumais, S., Fuhr, N., Harman, D., Harper, D. J., et al. (2003). Challenges in information retrieval and language modeling: report of a workshop held at the center for intelligent information retrieval, university of massachusetts amherst, september 2002. In *ACM SIGIR Forum*, volume 37, pages 31–47. ACM.
- Anastasiu, D. C., Gao, B. J., Jiang, X., and Karypis, G. (2013). A novel two-box search paradigm for query disambiguation. *World Wide Web*, 16(1):1–29.
- Backstrom, L., Huttenlocher, D., Kleinberg, J., and Lan, X. (2006). Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining (KDD)*, KDD '06, pages 226–231.
- Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., and Su, Z. (2007). Optimizing web search using social annotations. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 501–510, New York, NY, USA. ACM.
- Bennett, P., White, R., Chu, W., Dumais, S., Bailey, P., Borisyuk, F., and Cui, X. (2012a). Modeling and measuring the impact of short and long-term behavior on search personalization. In *Proceedings of the 35th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 185–194.
- Bennett, P. N., White, R. W., Chu, W., Dumais, S. T., Bailey, P., Borisyuk, F., and Cui, X. (2012b). Modeling the impact of short-and long-term behavior on search personalization. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 185–194. ACM.
- Carmel, D., Zwerdling, N., Guy, I., Koifman, S. O., Har'el, N., Ronen, I., Uziel, E., Yogev, S., and Chernov, S. (2009). Personalized social search based on the user's social network. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1227–1236, New York, NY, USA. ACM.
- Dou, Z., Song, R., and Wen, J. (2007). A large-scale evaluation and analysis of personalized search. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 581–590.

- Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
- Gantz, J. and Reinsel, D. (2011). Extracting value from chaos. *IDC iView*, pages 1–12.
- Goker, A. and Davies, J. (2009). *Information Retrieval: Searching in the 21st Century*. Wiley.
- Jansen, B. J., Spink, A., and Saracevic, T. (2000). Real life, real users, and real needs: a study and analysis of user queries on the web. *Information processing & management*, 36(2):207–227.
- Jeh, G. and Widom, J. (2003a). Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web, WWW '03*, pages 271–279.
- Jeh, G. and Widom, J. (2003b). Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM.
- Jiang, D., Leung, K., and Ng, W. (2011). Context-aware search personalization with concept preference. In *Proceedings of the 17th international conference on Information and knowledge management, CIKM '11*, pages 563–572.
- Liu, F., Yu, C., and Meng, W. (2002). Personalized web search by mapping user queries to categories. In *Proceedings of the 8th international conference on Information and knowledge management, CIKM '02*, pages 558–565. ACM.
- Ma, Z., Pant, G., and Sheng, O. (2007). Interest-based personalized search. *ACM Transactions on Information Systems (TOIS)*, 25:5:1–5:25.
- Maeda, A., Sadat, F., Yoshikawa, M., and Uemura, S. (2000). Query term disambiguation for web cross-language information retrieval using a search engine. In *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 25–32. ACM.
- Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, MA.
- Matthijs, N. and Radlinski, F. (2011). Personalizing web search using long term browsing history. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 25–34. ACM.

- Micarelli, A., Gasparetti, F., Sciarrone, F., and Gauch, S. (2007). Personalized Search on the World Wide Web The Adaptive Web. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 6, pages 195–230. Springer Berlin / Heidelberg, Berlin, Heidelberg.
- Miller, G. A. (1995). WordNet: a lexical database for English. *Commun. ACM*, 38(11):39–41.
- Noll, M. and Meinel, C. (2008). Web search personalization via social bookmarking and tagging. pages 367–380.
- Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: bringing order to the web.
- Pretschner, A. and Gauch, S. (1999). Ontology based personalized search. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, ICTAI '99*, pages 391–398, Washington, DC, USA. IEEE Computer Society.
- Qiu, F. and Cho, J. (2006). Automatic identification of user interest for personalized search. In *Proceedings of the 15th international conference on World Wide Web, WWW '06*, pages 727–736.
- Radlinski, F. and Dumais, S. (2006). Improving personalized web search using result diversification. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '06*, pages 691–692, New York, NY, USA. ACM.
- Sharifi, B., Hutton, M.-A., and Kalita, J. (2010). Summarizing microblogs automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 685–688. Association for Computational Linguistics.
- Shen, X., Tan, B., and Zhai, C. (2005). Implicit user modeling for personalized search. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831, New York, NY, USA. ACM.

- Sieg, A., Mobasher, B., and Burke, R. (2007). Web search personalization with ontological user profiles. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, CIKM '07*, pages 525–534, New York, NY, USA. ACM.
- Simon, H. (1971). Designing organizations for an information-rich world. *Computers, communications and the public interest, Johns Hopkins Press, Baltimore*, pages 38–52.
- Speretta, M. and Gauch, S. (2005). Personalized search based on user search histories. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 622–628.
- Sugiyama, K., Hatano, K., and Yoshikawa, M. (2004). Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM.
- Sun, J., Zeng, H., Liu, H., Lu, Y., and Chen, Z. (2005). Cubesvd: A novel approach to personalized web search. In *Proceedings of the 14th international conference on World Wide Web, WWW '05*, pages 382–390.
- Teevan, J., Dumais, S. T., and Horvitz, E. (2005). Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456. ACM.
- Ugander, J., Karrer, B., Backstrom, L., and Marlow, C. (2011). The anatomy of the facebook social graph. *CoRR*, abs/1111.4503.
- Zamir, O. and Etzioni, O. (1998). Web document clustering: A feasibility demonstration. In *Proceeding of the 21th international ACM SIGIR conference on Research and development in information retrieval, SIGIR '98*, pages 46–54, New York, NY, USA. ACM.