MOBILE WEB SEARCH PERSONALIZATION USING ONTOLOGICAL USER PROFILE

by

KAPIL GOENKA

(Under the Direction of I. Budak Arpinar)

ABSTRACT

Most present day search engines have a deterministic behavior in the sense that they return the same search results for all users who submit the same query at a certain time. They do not take the user's interests and preferences into account in the retrieval process. Integrating user context in the retrieval process can help deliver more targeted search results, thereby providing a personalized search experience to the user. Personalizing web search involves the process of identifying user interests during interaction with the user, and then using that information to deliver results that are more relevant to the user. In this thesis, we present our approach to personalizing web search on a mobile device (iPhone). Our approach involves building an ontological model of user interests on the user's mobile device based on his interaction with web search results. Personalization of search results is achieved by re-ranking search results returned by a standard search engine (Yahoo) based on proximity to the user's interest model. The ability to recognize user interests in a completely non-invasive way and the accuracy of personalized results are some of the major advantages of our approach.

INDEX WORDS: Search Personalization, User Profiles, iPhone

MOBILE WEB SEARCH PERSONALIZATION USING ONTOLOGICAL USER PROFILE

by

KAPIL GOENKA

B.E., Birla Institute of Technology & Science, Pilani, India, 2007

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillments of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2009

© 2009

KAPIL GOENKA

ALL RIGHTS RESERVED

MOBILE WEB SEARCH PERSONALIZATION USING ONTOLOGICAL USER PROFILE

by

KAPIL GOENKA

Major Professor :	I. Budak Arpinar
Committee:	John Miller Krzysztof Kochut

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2009

TABLE OF CONTENTS

Page

CHAPTER

Chapter 1

Introduction

Today, internet search engines have become an indispensable part of our lives. They have enabled mass participation and collaboration by hundreds of millions of people around the world. People today are able to find all sorts of information instantly from almost anywhere. Search engines have also come to be included within large web sites such as e-commerce sites, corporate sites, and social networking sites. The exceedingly difficult nature of the problem of understanding user intent and matching it with the world's accumulated knowledge stored on the World Wide Web has attracted large scale research and development efforts from the academia as well as the industry.

In the recent years, we have also seen an explosive growth in mobile devices. The modern cell phones are significantly better than the one's from a few years ago. They are more powerful, provide a much richer user experience and provide users with ubiquitous access to information more than ever before. Mobile Internet has quickly become part of the consumer media experience for millions of people. Unlike early adopters that originally used smart phones primarily for business, most new smartphone owners are using them for mostly personal use [22]. "As of May 2008, 40 million mobile subscribers in the US, plus millions more across the world, surfed the web through a mobile phone each month — checking email, exploring their social networks, making bank transactions and engaging in other web activities right from their hands" [17]. More and more people are searching the web while they are on the move. For instance, one might want to find information about a local restaurant, or need to learn more about a

new place they are visiting. The location-awareness of the modern mobile devices can help address such queries. DBPedia Mobile [23] is one such location aware application for the iPhone that uses the GPS location of the client to display a map of user's location. The user can select points of interest on the map and learn background information about them, navigating the many inter-connected linked data sources.

Although the capabilities of Internet search engines are incrementally improving, there are several challenges facing the search engines. One challenge is the problem of irrelevant search results. Irrelevant search results usually arise due to short, ambiguous queries or semantic level mismatches. Examples include "apple", "Pascal", "match", "conductor" etc. all of which can have different meanings depending on context. Another cause for irrelevant results is the one-size-fits-all approach taken by most existing search engines, where an identical query from different users in different contexts will generate the same set of results for all users. These search engines return a list of search results based on a user's query but ignore the user's specific interests, search context and individual differences in information needs. As a result, a user may have to go through many irrelevant results before finding the desired information. Problems encountered in searching are exaggerated further when search engine users employ short queries [1]. *Polysemy* - existence of multiple meanings for a single word, and *synonymy* - existence of multiple words with the same meaning, are two other problems that the keyword-based search approaches suffer from [18]. They cause relevant information to be missed if the query does not contain the exact keywords occurring in the documents. For these reasons, users face a difficult battle when searching for the exact documents and products that match their needs.

Mobile web search introduces new challenges not present in traditional web search. The input modes are inherently limited due to the small size of the device itself and the network connectivity is often not comparable to the Internet speeds on computers. Mobile users are likely to be on the go when searching for information and the attention span of the users is significantly lower than in traditional web search on computers. Furthermore, the user is unlikely to sift through a lot of search results to get to the desired page due to his short attention span. It is therefore very important to get the desired search results in the top positions to avoid waste of time and effort for the user.

As discussed above, the effectiveness of search technologies is reduced by the ambiguity of the user's query and the diversity of their information needs. Personalization techniques that incorporate user interests and preferences into the search may address some of these issues. Personalization broadly involves the process of learning a profile of user interests. It is then used to deliver personalized content to the user. Personalization of web search usually involves filtering or re-ranking the results returned from a standard search engine, or directly incorporating user interests into the retrieval process itself to present personalized results. Given a query, a personalized search can provide different results for different users or even different results for the same user in different contexts.

Web search personalization has two main dimensions:

1. How can precise information about user's interests be collected and represented?
2. How can this information be used to deliver personalized search results?

In this work, we present our approach to personalizing web search in a mobile environment. As a case study, we chose Apple's iPhone as the mobile platform to implement our work. Our main goal is to identify user's interests based on the web pages he visits, and deliver personalized web search results by utilizing the identified user interests. We learn and maintain implicitly an ontological profile of user's interests through passive observation of the user's clickstream. The user's interest profile is stored locally on his mobile device and updated with every web page visit. Personalization is achieved by re-ranking standard web search results using the user's interest profile.

Chapter 2

Background

2.1 Web Directories

Web Directories, also referred to as *Internet Directories* or *Knowledge Bases*, are a popular means of organizing information resources on the web. A web directory is a repository of web pages that are organized in a hierarchical structure, usually like a tree or a directed acyclic graph (DAG). Each web page cataloged in a web directory is annotated with a short description by one of the editors of the directory. The assumption behind the hierarchical structure is that each node in the hierarchy is a special type of its parent node and a general type of its child nodes, thereby implying a hierarchical relationship. Each non-leaf node in the hierarchy defines a *concept,* and each leaf node defines a list of web pages (along with descriptions) that have been cataloged under the parent of the leaf node by the web directory editors. Every concept node 'C' except the root has a parent node, may have any number of child nodes representing its sub-concepts and may have at most one child node. In DAG-structured web directories, a concept node may have more than one parent nodes. Typically, directory users locate information in a web directory by browsing through the concept hierarchy, identifying the relevant concepts and finally examining the pages listed under the relevant concepts.

Many web directories have become available in recent years. The Librarian's Internet Index (LII) [3], The Internet Public Library (IPL) [4], Yahoo Web Directory [5] and the Open Directory Project (ODP) [6] are examples of general purpose web directories. These web directories catalog huge numbers of URLs organized in an elaborate

hierarchy. Since the actual process of creating such ontologies can be a very tedious, most hierarchical classification systems utilize existing web directories as their predefined class hierarchies.

2.1.1 Open Directory Project (ODP)

In this work, we use the Open Directory Project (ODP) as our knowledge base. ODP is one of the largest collaborative efforts to manually annotate web pages and is widely regarded as “the largest human-edited directory of the web”. Currently ODP catalogues over 4.6 million URLs that have been categorized into nearly 600,000 categories by over 80,000 human editors. ODP’s data structure is organized as a DAG. The textual data contained in the leaf nodes can be utilized as training data for the parent concept of the leaf node. Since ODP truly is free and open, everybody can contribute or reuse the dataset, which is available in RDF format.

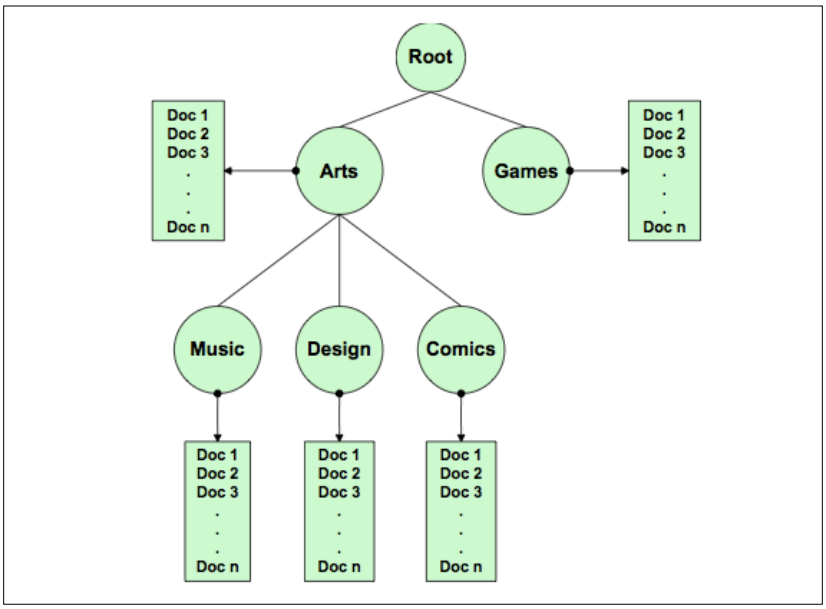


Fig 1 : A portion of ODP hierarchy. Circular nodes represent concepts, rectangular nodes represent leaves

Besides its reuse in other directory services, ODP is useful as a basis for various other research projects. Google for example uses ODP as the basis of its Google Directory service. ODP can be used as a web corpus for comparison of rank algorithms [8], as well as for focused crawling towards special-interest pages [9, 10]. ODP has been applied to personalization of web search in some prior studies [11].

Web directories like ODP cover most, if not all, information domains, and can therefore be used for representing user interests. Nodes at the top levels of the hierarchy represent broad user interests and the ones below them narrow down the scope of their ancestors. In this work, we select a subset of concepts from the top four levels of ODP for representing user interests. Chapter 4 contains a discussion of our intuitions behind selecting the subset of concepts. We focus on the top levels of the hierarchy since we believe that many search results can be usefully disambiguated at this level.

2.2 Text Classification

2.2.1 The Basic Picture

Text classification (also known as *text categorization*) is the task of automatically sorting documents into categories from a predefined set. Text classification is particularly useful for managing large bodies of information. The problem of text classification has been well studied in the past. This section takes a closer look at text classification, by giving a basic picture of how automated text classification systems (*classifiers*) are built and tested. The categories in text classification are just labels. Usually, no additional knowledge of their meaning is available except for the documents indexed under them. In these cases, learning must be accomplished based on the documents available.

Text classification is executed in two steps: *training* and *classification*. In the training step, the system is given a set of training documents, which has examples for which documents go into which categories. From this, the system learns a model of the information contained in each of the categories. In the classification phase, the system receives a new document and assigns it to a particular category, based on matches between its features and those extracted from the training documents. Several methods for document classification have been developed. A survey and comparison of such methods is presented in [12].

2.2.2 Types of Classifiers

Single-label vs Multi-label:

Text classification may be either a *single-label* task (in which a new document must be assigned to exactly one category), or a *multi-label* task (in which a document may be assigned to any number of categories). The problem of multi-label classification is often broken down into independent binary classification problems. This is also true for many hierarchical classifiers. These classifiers train a binary classifier at each node of the hierarchy.

Flat vs Hierarchical:

Classification may also be *flat,* where the relationship between the categories is undetermined, or *hierarchical*, where the categories are part of a hierarchical structure, such as a tree or a Directed Acyclic Graph (DAG). A vast majority of the previous work on classification has focused on flat classifiers, although recently several researchers

have investigated the use of hierarchies for text classification, with promising results. By utilizing known hierarchical structure, the classification problem can be decomposed into a set of smaller problems corresponding to the hierarchical splits. Each sub-problem is smaller than the original problem, and in the lower levels of the hierarchy, it is possible to use a much smaller set of features for training a classifier.

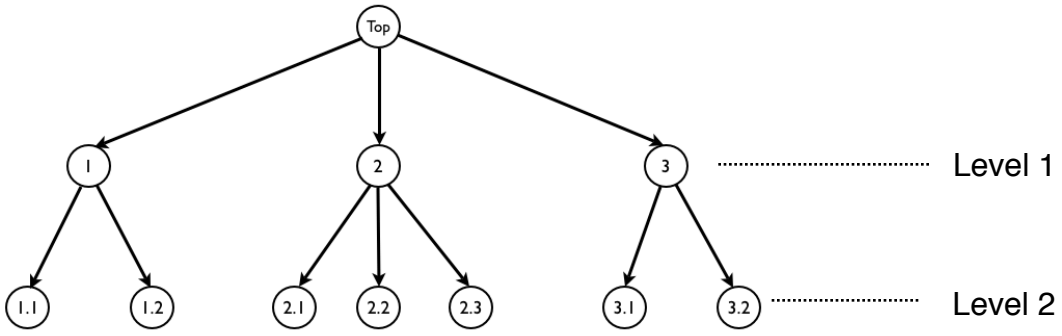


Fig 2: Hierarchical structure

The problem of hierarchical classification is usually tackled by training a different classifier for each node of the class hierarchy. Further, a threshold is usually specified. A category is considered a match for a given document only if the classifier assigns a score above the threshold to the category for the given document. A new document is classified first at the top level of the hierarchy. The document is then sent down the hierarchy along the path of category matches until the either the document reaches a leaf or classification at a given level cannot be performed with accuracy above the threshold.

While hierarchical classifiers have been shown to achieve modest to good accuracy improvement over flat classifiers, hierarchical classification model is not suitable for our research. This is because in the hierarchical case, a search result must be classified by

multiple classifiers before arriving at the final class assignment. Based on our experiments, it takes somewhere between 500 ms to 750 ms in most cases to classify the top 100 Yahoo search results at any given level of the hierarchy. Assuming we build a hierarchy classifier that operates only on the top 3 levels of ODP, it would still take around 2 seconds, and that too if we just considered exactly one class assignment at each level.

2.2.3 Text Classification Using Rainbow

We use the open source Rainbow text classification library [21] by Andrew McCallum at CMU as the "kernel" of our text classification module. The Rainbow Text Classifier, is perhaps the most well known and most downloaded text classifier today. It supports a number of text classification methods for classifying text into a set of topics. Rainbow must be trained. before using it for classification. This involves creating a model of a set of training documents. The training set is read in as directories (one per category) containing text files that serve as examples for those categories. Once Rainbow is trained, it can be set up as a server that received classification requests over a port.

Feature selection is known to improve performance of text classification by reducing the feature set or vocabulary size. Rainbow can conduct feature selection using one of three approaches: removing words that occur in N or fewer documents; words that occur less than N times; or by removing all but the top N words by selecting words with highest average mutual information with the class variable. Rainbow also provides some diagnostic options. For example, we can get a list of the top N words in terms of mutual information with the classes (see Table 1). After a model is learnt from the

training set, classification can be performed using one of the many classification methods supported by Rainbow - Näıve Bayes, Term Frequency - Inverse Document Frequency (TFIDF), probabilistic indexing, k-nearest neighbor and support vector machines (SVMs).

ODP Concept	Top 10 selected attributes
Circus	juggling, circus, cirque, fire, trapeze, soleil, jugglers, clown, juggler
Dentistry	dental, dentistry, oral, dentists, health, education, association, orthodontic, teeth, treatment
Dictionaries	dictionary, terms, english, glossary, words, dictionaries, definitions, terminology, word, acronyms
Halloween	halloween, ideas, party, decorations, spooky, snacks, decorating, invitations, costumes, haunted
Pink_Floyd	floyd, pink, lyrics, band, tribute, wall, discography, album, albums, cover
Programming	programming, code, software, development, uml, java, source, powerbuilder, tools, open
Thanksgiving	thanksgiving, turkey, recipes, cookies, holiday, mayflower, menu, cranberry, turkeys, intensive
Zoology	zoo, animal, animals, species, park, conservation, information, research, aquarium, visitor
Speech_Technology	speech, recognition, voice, solutions, text, synthesis, software, iv, dictation, applications
Concurrent_Programming	threads, thread, programming, synchronization, java, concurrency, posix, lock, concurrent, multithreaded

Table 1 : Examples of attribute selection using information gain

In this work, we train the rainbow classifier on a subset of the first four levels of the ODP categories and set the classifier to be run as a continuous background server

process. The classifier listens for document classification requests over a port and produces a classification score for each category for which it was trained.

2.3 Web Search APIs

Many popular commercial search engines (Google, MSN and Yahoo) have developed freely available APIs for accessing their index. Google started offering a free SOAP based API for accessing its index since early 2002. Yahoo and Microsoft released public APIs in 2005 as well. Prior to the release of the search engine APIs, “focused crawling” – crawling the Web for content of interest was a popular way to build a digital library. By opening up their search indexes, the search companies enable researchers & search startups to overcome obstacles such as cost (it would take a lot of effort, time and money to build search engines like Yahoo from scratch) and empower them to further innovate and improve web search.

2.3.1 Yahoo BOSS API

Yahoo BOSS (Build Your Own Search Service) is an open platform that offers programmatic access to the Yahoo Search indices via an API. As of this writing, the Yahoo BOSS API is offered free of charge to developers. There is no limit to the number of queries that can be made. However, a maximum of 50 search results can be fetched per query. The search API allows developers to specify the start position of search results. So fetching the top 500 search results for a query would involve sending 10 API requests, starting with a start position of 0 and incrementing the start position by 50 with each request. We use the BOSS Mashup Framework [25] -- a Python library provided by Yahoo to access the Yahoo search results. We note that other search engine APIs

can be used for retrieving standard search results, in pace of Yahoo API. We decided to use the Yahoo API mainly because it provides us access to “key terms” for each search result.

Key terms are keywords Yahoo's search index has assigned to a page. It is a finite list of words that explain what a document is about and allow for better categorization. The key terms are obtained by Yahoo based on each term’s frequency & positional attributes in the document. Key terms are particularly useful in our work, as they save us valuable post-processing time which would otherwise be required for processing result pages & obtaining the key words representing each page. For the purpose of classifying web search results, we consider the combination of key terms, title and snippet of each search result as sufficient information for representing what the web page is about.

Fig (3) illustrates key terms corresponding to the first search result for a sample query of [obama].

Fig 3 : Key terms for a Yahoo search result

Chapter 3

Web Search Personalization

This chapter reviews prior studies pertaining to personalized search. Many projects aim at improving web search by taking into account the interests of each user.

3.1 Related Work

In [13], they use the search history of each user to assign each user to a set of categories. When this user submits a query, the search engine will add his categories to the query to modify the submitted representation of the information need. In our research, we use the results re-ranking approach for personalization rather than the query modification approach. In [14], Hattori et al. consider web search from a mobile device. The user's geographical location and context are used to refine queries submitted by the user. In [15], they monitor users' activities by capturing content from Internet Explorer and Microsoft Word applications. Given that user typically spend a lot of time interacting with the web browser and their text editing applications, it is a good approach to monitor changing user interests. Stuff I've Seen [16], developed at Microsoft Research, indexes all the content seen by a user. The index is later used to provide easier access to information already seen by the user, and also to personalize information retrieval.

3.2 Approaches To Search Personalization

3.2.1 User Modeling

A user model stores an approximation of user's interests. User models are used to personalization systems to tailor generic content to the particular needs of a user. User

models are often updated automatically by tracking user's clickstream, websites visited, etc. An example of a simple personalized information discovery tool is the Google Alerts system [31]. Users specify explicitly what they are interested in, for instance a phrase or keywords. The system notifies users when new information is published on the web that contains their phrase and/or keywords. In more complex systems, the user model is dynamic and adapts over time to reflect changes in user interests.

In personalized search systems the user modeling component can affect the search in three distinct phases, showed in Fig (4):

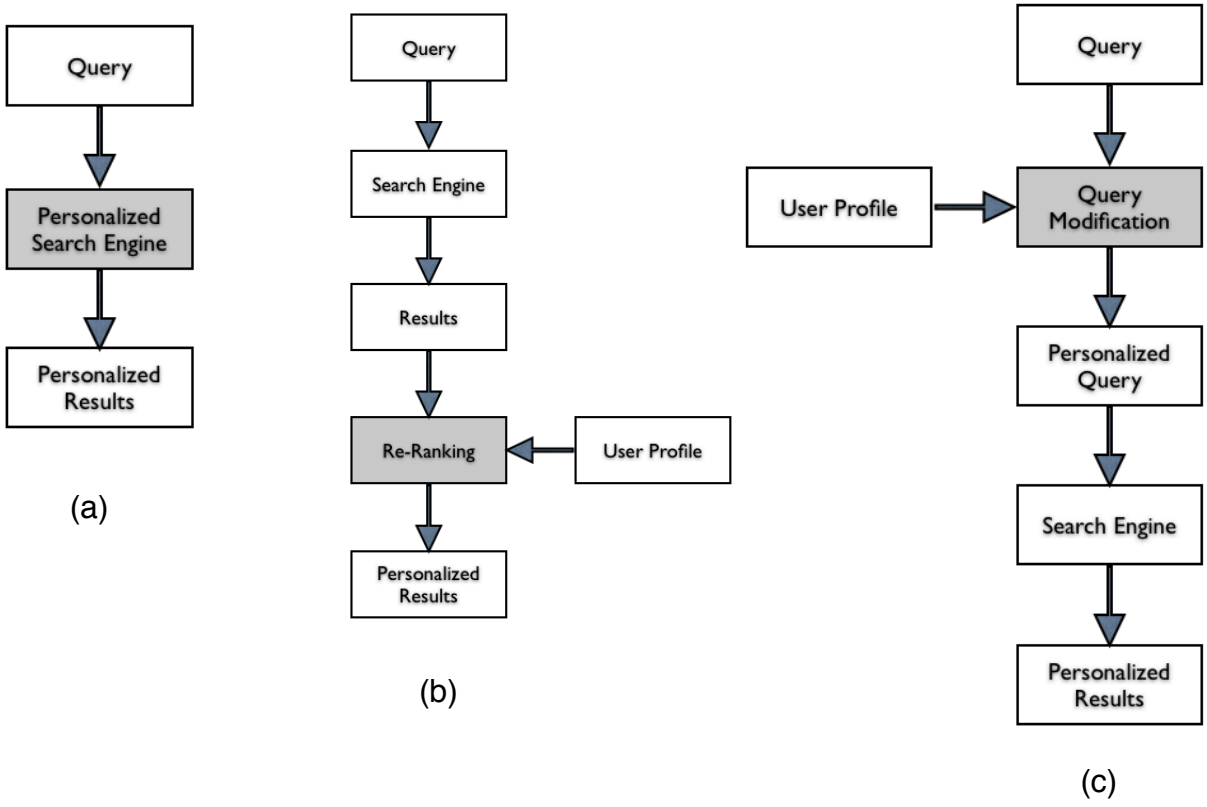


Fig 4 : Personalization process where the user profile occurs (a) During the retrieval process (b) in a distinct re-ranking activity or (c) in a pre-processing of the user query

- part of retrieval process: user profiles are built into the search process, and are used to score web documents. The search engine is designed with personalization in mind. This method and does rely on an external search that provides search results.
- re-ranking: this method receives content (search results) from a standard non-personalized search engine, and personalizes the content in a second step by taking into account the particular user's interests.
- query modification: In this approach, user profiles are used to modify the submitted representation of the information needs.

Given the time constraints forced on the search systems, and the fact that personalization is a slow process and user profiles only get better with time and use, most search engines do not employ any personalization at all.

Personalization systems that re-rank the documents returned from a standard retrieval process often employ user profile on the client-side. Moreover, instead of getting all results from the source, they typically get top ranked documents and re-rank them. Because of the time needed for the additional re-ranking step this approach can be considerably slow. Nevertheless, complex user needs can be employed, and high level of personalization can be achieved.

In the query modification approaches, user profiles affect the ranking only by altering the query representations. Query modification is therefore less likely to affect the result lists, because it does not have access to all the ranking process.

3.2.2 Collaborative Filtering

The concept of collaborative filtering is to provide recommendations to a user based on previous ratings by users with similar interests and background. Collaborative filtering can help users discover popular content in the fields on their interests. Moreover, this type of personalization often does not depend on the content of the items, and there works well for non-textual contents like music, movies and images.

3.2.3 Result Clustering

The idea behind result clustering is to group search results into clusters. Each cluster contains search results related to the same topic. Clusters provide an overview of the retrieved results to the user and may be considered a good starting point. Users are able to navigate the appropriate cluster based by their search needs.

Chapter 4

Methodology

In chapters 2 and 3, we gave some background about the Open Directory Project [6], text classification, Rainbow text classification library [22] and building user profiles for personalizing web search. In this chapter, we present our methodology to put these components together for personalizing web search on a mobile device. Section 4.1 describes our approach for programmatically accessing ODP and replicating its structure on a local file system. Section 4.2 describes the steps we take to remove structural noise from ODP. Section 4.3 is about training a text classifier based on ODP. Section 4.4 describes our system software architecture and how we use the Yahoo BOSS Mashup Framework to retrieve web search results in a structured format. Section 4.4.1 describes the server-side part of our system and section 4.4.2 describes the client-side part, in which we present our method for building user profile on the user's iPhone and describe how the user profile can be used to re-rank search results.

4.1 Programmatically Accessing ODP

ODP provides two separate RDF files: *structure.rdf* which includes the category hierarchy information; and *content.rdf* which contains all the related information of each category and links with each category. We start with the MySQL database dump of the these two RDF files, published in [1]. The MySQL dump provides us a convenient SQL interface to the entire ODP hierarchy. The database contains several tables that together capture all the information in the ODP hierarchy. We're interested in topics (categories) and web pages indexed under topics. This information is contained in the

ttopics, ttopiclinks, ttopicnarrow and textLinks tables of the database. The schemas for these tables are shown below.

Table Schema : ttopics

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	YES	UNI	NULL	
sPath	varchar(255)	YES	UNI	NULL	
sDispname	text	YES		NULL	
sTitle	text	YES		NULL	
sDescription	text	YES		NULL	

Table Schema : ttopiclinks

Field	Type	Null	Key	Default	Extra
idTopic	int(10) unsigned	YES	MUL	NULL	
idType	int(10) unsigned	YES		NULL	
idExtLink	int(10) unsigned	YES		NULL	

Table Schema : ttopicnarrow

Field	Type	Null	Key	Default	Extra
idTopic	int(10) unsigned	YES	MUL	NULL	
idType	int(10) unsigned	YES		NULL	
idNarrowTopic	int(10) unsigned	YES		NULL	

Table Schema : textLinks

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto_increment
idProtocol	tinyint(3) unsigned	YES		1	
idHost	int(10) unsigned	YES		NULL	
sPage	text	YES		NULL	
sTitle	text	YES		NULL	
sDescrip	text	YES		NULL	

The **ttopics** database table contains the path and title information of every topic in ODP. For example, the ‘Chat’ topic is categorized under ‘Internet’ which is under ‘Computers’. So the path of the ‘Chat’ topic is Top/Computers/Internet/Chat and its title is ‘Chat’.

The **ttopicnarrow** database table captures the hierarchical structure ODP. Given a topic, we can get a list of all its sub-topics from the ttopicnarrow table. For example, the id of the Top/Regional/Asia topic is 262; so the list of all its sub-topics can be obtained from the ttopicnarrow table using the SQL query:

```
select idnarrowtopic from ttopicnarrow where idtopic = 262;
```

The **ttopiclinks** database table contains the ids of the webpages categorized under each topic. Once we obtain the ids of the webpages, we can then lookup the **textLinks** database table to get the Title, URL and description of the webpages.

We query the database using a Java program that connects to the database through a MySQL JDBC connector. Using the Java program, we create a directory structure on the local file system that replicates the ODP hierarchy. The procedure for doing this is shown below. The output of the procedure is a directory structure where each directory represents a concept, with the property that the fully qualified path from the top of the directory structure to the current concept matches the concept’s path in ODP. Each directory contains subdirectories representing its sub-concepts, and a Super Document containing the title and description of every web page categorized under that concept.

PROCEDURE: CREATE_ODP_STRUCTURE

OUTPUT: A directory structure on the file system replicating the ODP metadata

For every category C in the ODP hierarchy:

    path <--- path of C in the ODP hierarchy

    numSubTopics <--- Number of subcategories of C

    numLinks <--- Number of web pages categorized under C

    if numSubTopics == 0 AND numLinks == 0

        Ignore category C

    else

        Create a directory for category C at location specified by variable 'path'

        if numLinks > 0

            Create Super Document SD_c

            For every web page W categorized under C:

                Add the Title & Description of W to SD_c

            Save SD_c in C's directory

Fig 5 : Procedure to replicate ODP structure on a local hard drive

4.2 Removing Structural Noise from ODP

However elaborate knowledge repositories are, they contain concepts that are detrimental to feature generation [20]. These include concepts too deep in the hierarchy, or having too few textual objects to build a representative attribute vector.

In [20], they have identified potential sources of noise in ODP. In our work, we use their findings to prune the following topics from ODP:

1. The branch Top/World concentrates on material in languages other than English.
2. Top/Adult lists adult-oriented Web sites, and we believe that the concepts of this subtree are of little use for general purpose text categorization.
3. Top/Kids_And_Teens roughly duplicates the structure of the ODP but only lists resources suitable for children.
4. The Top/Regional branch is devoted to listing English language sites about various geographical regions of the world. Regional concepts often contain similar features (e.g., 'county', 'district', 'state'), which are not discriminating enough. Such classes affect the accuracy of a multi-label classifier, because when classifying a test document that is about a geographical region, the classifier often ends up matching the incorrect category.

The above four branches are, therefore, completely pruned.

4.3 Training a Text Classifier on ODP

As discussed in section 2.2, we use the Rainbow text classification library to train a flat multi-label text classifier on a subset of categories from the top four levels of ODP. We cannot simply flatten the top four levels of ODP and use all the categories for training the classifier. This is because flattening breaks the parent-child relationship between the categories and brings them to the same level. After flattening the top four levels, we therefore carefully remove all the categories that do not make sense individually. There is no algorithmic way of doing this, we use our best intuition and experimental results to arrive at the final set of categories that will represent the user interests.

In the training phase, Rainbow reads in a set of directories (one per category) containing text files that serve as examples of those categories. It then builds a statistical model of the corpus, which is stored on disk. Once trained the classifier can be used to classify new documents. Upon classifying a document, Rainbow produces a classification score for each category on which it was trained. The classification score is between 0 and 1, and is a measure of the match between the document and the category, 1 being an exact match. The classifier is central to our personalization system. Sub-optimal classification results will lead to an inaccurate user model, which may eventually cause irrelevant search results to be returned to the user. We therefore designed a number of experiments around the text classifier. These experiments and their results are shown in chapter 5. In this section, we present and compare two different approaches we took for training a document classifier based on ODP.

APPROACH I:

1) Select a subset 'S' of concepts from ODP to be used in user modeling.

S = {C1, C2, C3, C4, C5, C6 ..... Cn}

2) Since the concepts in S are from different levels in ODP, flatten them, i.e., move them (along with their sub-trees) to a common level. This is done because we need to train a flat classifier over the categories in S.

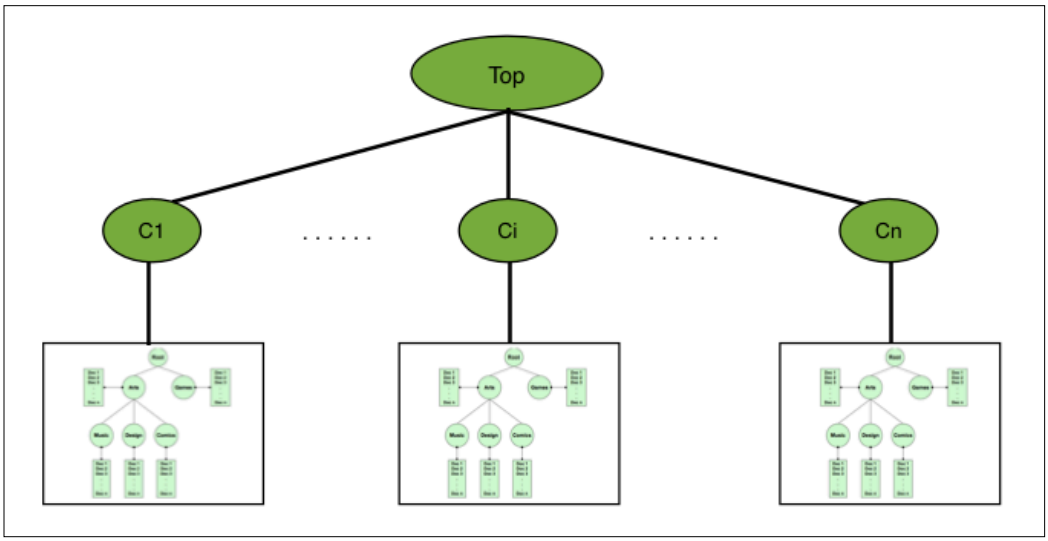


Fig 6 : Flattening the categories selected from top four levels of ODP

3) Train the text classifier, using all textual documents under a concept as the training data for that concept.

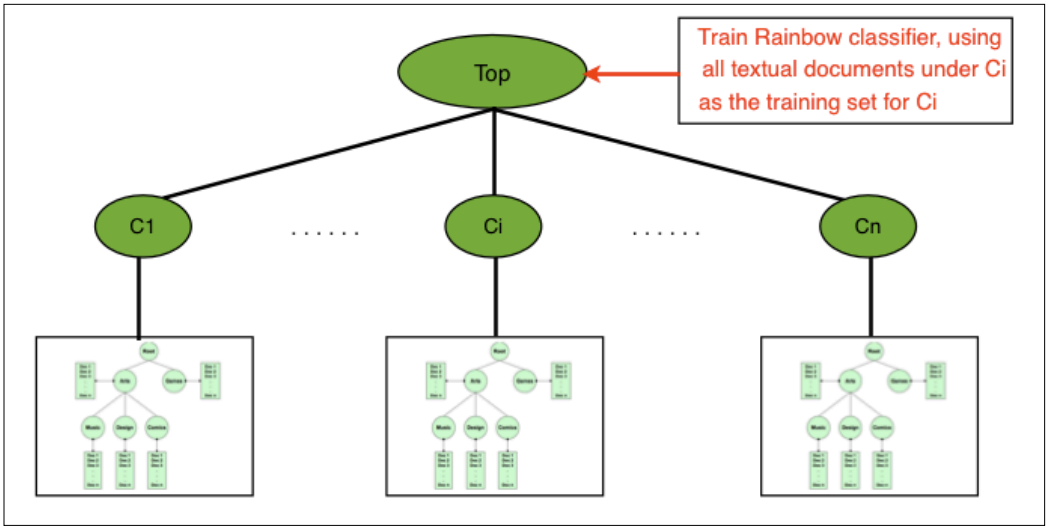


Fig 7 : Training a Rainbow classifier on the categories

4) Set up Rainbow to receive classification requests on a specific server port.

DISCUSSION of APPROACH I:

Since we use all documents for training purpose, the number of features per category is very large. A large feature set leads to a performance loss in many cases. Moreover, since different classes have different amounts of textual data under them, classes with larger amounts of textual data appear much more often in classification results as compared to the ones with smaller amounts of textual data. This is because the classification results are based on word probabilities and occurrence counts which creates a bias towards the classes with more data. This clearly leads to a sub-optimal quality of classification results, and in turn a lower quality of the system generated user profile. To overcome the data imbalance problem, we need a way to 'equalize' the classes and reduce the feature set.

APPROACH II:

Steps 1) and 2) as in APPROACH I.

3) Run Rainbow document classifier at the level 'Top', this time indexing only 20 randomly selected documents under each class. Selecting the same number of documents for each class overcomes the data imbalance problem of APPROACH I.

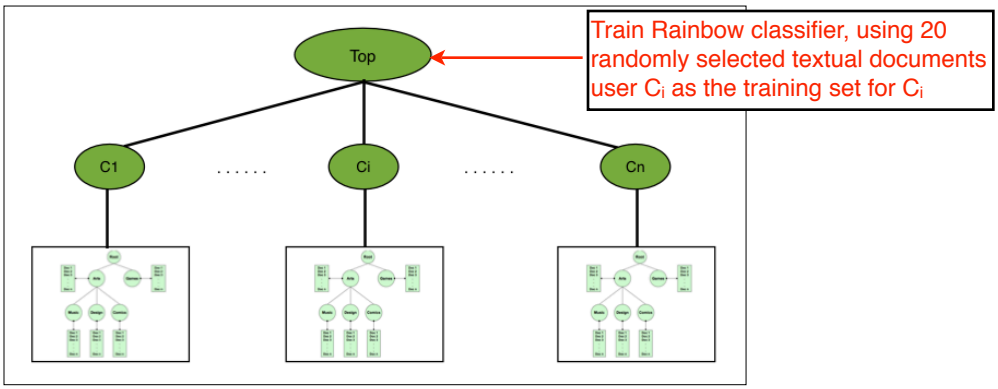


Fig 8 : Training a Rainbow classifier on the categories

4) Set up Rainbow as a server on a specific port.

DISCUSSION OF APPROACH II :

In this approach, we intend to reduce the feature set of the TF IDF based classifier. Feature selection in text classification has been repeatedly shown to lead to little accuracy loss, and to a performance gain in many cases. Our method of reducing the features is to select a smaller, fixed number of training documents per category. Selecting a fixed number of training documents per category equalizes the categories, and since each of training documents contain rich textual information about a number web pages, selecting even a small number of training documents per category results in a rich feature set.

4.4 System Software Architecture

In this section, we describe the software architecture of our personalization system. Fig (9) gives an overview of the system.

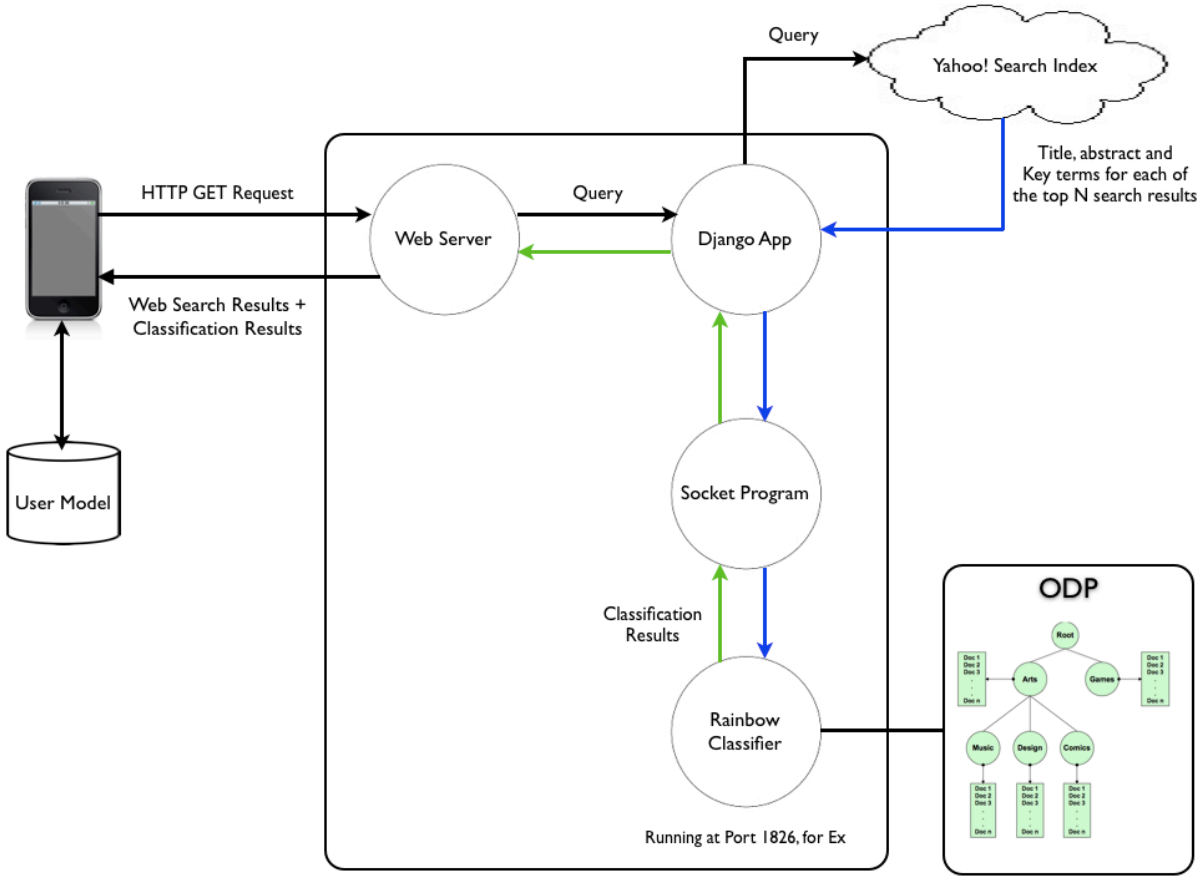


Fig 9 : System Architecture

A good way to understand the working of our system is to view it as being composed of two parts - i) the client-side part which resides on user's iPhone, and ii) the server-side part which is implemented on a server. Section 4.4.1 describes the server-side part of our system, and section 4.4.2 describes the client-side part.

4.4.1 Server-Side

The server-side of our system consists of three main components:

i) A text classifier, trained as described in section 4.3

ii) A socket program that communicates with the text classifier over a server port.

iii) A Django application that receives search query from the user, retrieves Yahoo search results for the query, forwards them for classification and returns the search results along with their classification back to the client device.

Django Application:

Django is an open source web application framework, written in Python. Django can be run in conjunction with Apache using mod_python. mod_python is an Apache HTTP Server module that integrates the Python programming language into the Apache server. The 'Django App' component of fig (4) is an integral server-side component of our system. It integrates with the Yahoo BOSS search framework. Specifically, the Django App receives search query from the client device and retrieves Yahoo search results for the query using the "BOSS Mashup Framework". It then sends the search results to the 'Socket Program' component, and receives the classification results back from the 'Socket program' component. Finally, it sends the results back to the user.

Socket Program:

The Socket Program component in fig (4) is a C program that performs socket communication. We set our Rainbow classifier to be run as a continuous background server process. The classifier listens for document classification requests over a port and produces a classification score for each category for which it was trained. The

socket program does the job of sending document classification requests to the port on which Rainbow is running, and reading the document classification result scores back from the port.

Whenever the user performs a search on his iPhone, an HTTP GET request containing the user query is sent to our web server. The web server is configured to forward such requests to the Django application. The Django application receives the HTTP request URL from the web server and extracts the query from the URL. It then performs Yahoo web search for the query through the BOSS API. We fetch the top 100 search results from Yahoo. That corresponds to the first 10 pages of search results. Given that users typically browse up to the top 2 to 3 result pages on an average, we believe that 100 search results will be reasonable in most cases. The search results are returned as a JavaScript Object Notation (JSON) formatted string. JSON is a lightweight data-interchange format that is based on a subset of the JavaScript Programming Language [27]. For each search result, the JSON string contains the Title, URL, abstract and key terms (among other data) corresponding to the web page. In our Django application, we combine the title, abstract and key terms for each search result into a single string. We believe that the combination of title, abstract and key terms for a web page provides sufficient information of what the web page is about. A more sophisticated approach would be to extract the complete text of the search result web page and analyze it to understand what the web page is about but that requires additional steps such as parsing out all the HTML content and performing text analysis, both of which add significantly to the post-processing time. Besides, the key terms were extracted by Yahoo by performing text analysis in the first place and provide much valuable

information about a search result web page in addition to its abstract. Therefore, using the Yahoo key terms is the same as performing text analysis on the web page content. In [26], they take a similar approach as ours wherein they used the Google SOAP API to access Google search results and used the search snippets as representing the search result web page.

4.4.2 Client-Side

In this section, we describe the client-side part of our system. As discussed earlier, we do not store any type of user information on the server. The user's interest profile is maintained locally on the user's iPhone. We model user interests using the same concepts that we trained our text classifier on.

Concept	Weight
C_1	W_1
C_2	W_2
:	:
:	:
:	:
C_n	W_n

Table 2 : User Profile on client side

Table 2 shows the structure of the user profile. The first column contains concept names and the second column contains concept weights in the user model. The concepts in

the user profile are the same as the ones on which the classifier is trained on the server side. Initially, all concept weights are zero. The concept weights are constantly updated by our system based on the user's interaction with the search results and based on the links the user visits after clicking one of the search results. Below, we give the details of how the weights are updated. At any time, the concepts with higher weights are the ones the user is more likely to be interested in.

When the user performs a web search, the request is sent to our server, which retrieves the search results from Yahoo, classifies them and sends search results along with their classification results back to the user's iPhone. Fig (10) below shows what information about each search result is returned from the server.

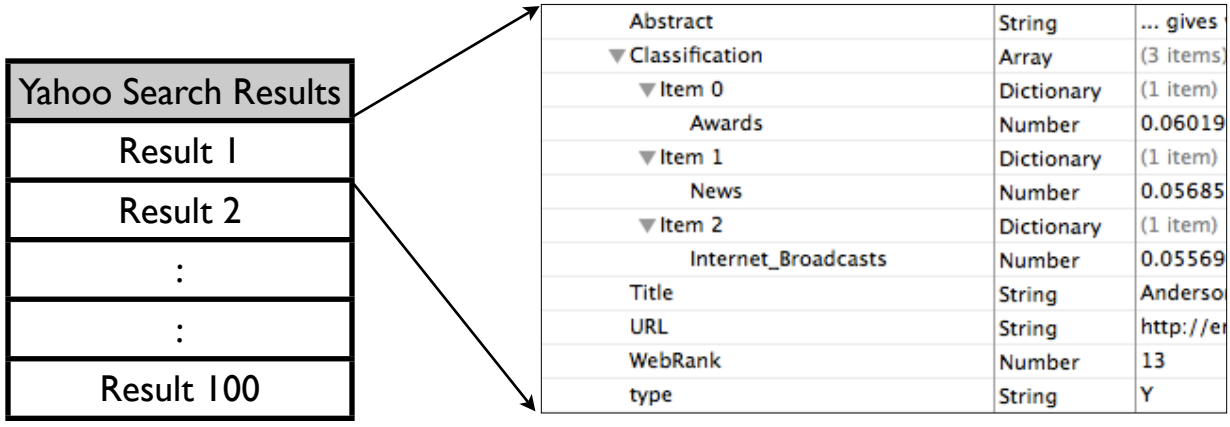


Fig 10 : Search Result details stored on the user's iPhone

For each search result, we return the Title, URL, Abstract, web rank and the top three categories assigned to that result by our document classifier.

Once the Yahoo search results are received, the next step on the client-side is to re-rank the results so that the ones that are more likely to be of interest to the user are shown above others.

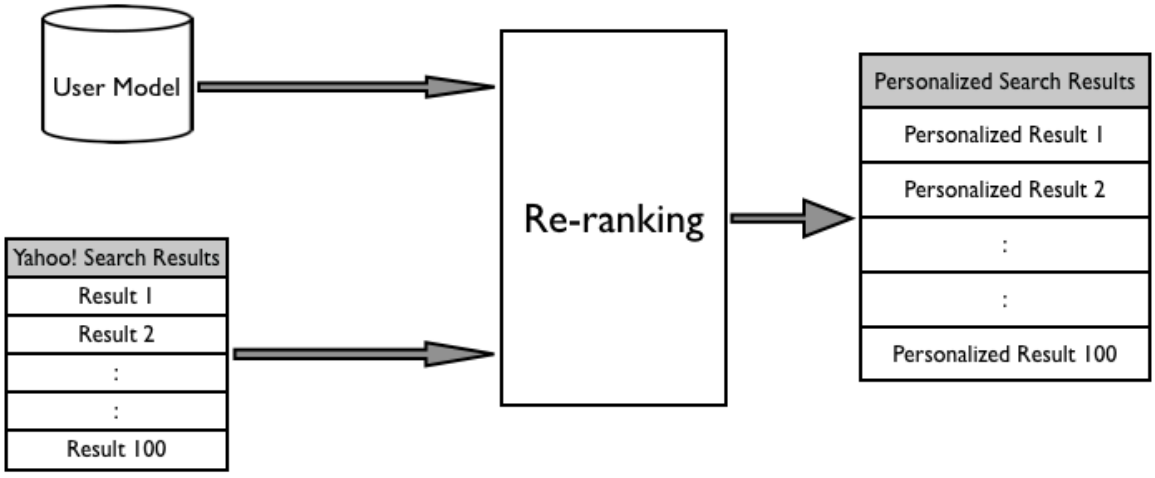


Fig 11: Re-ranking search results on the client side

The re-ranking is achieved through a matching function which calculates the degree of similarity between each search result and the user profile.

$$Sim(user_i, Result_j) = \sum_{k=1}^{N} wp_{i,k} \cdot wd_{j,k}$$

where $wp_{i,k}$ = weight of the concept k in the user profile i,

$wd_{j,k}$ = weight of the concept k in the result j,

N = number of concepts returned to the client

The final weight of the document used for reordering, so that the results that best match the user's interests are ranked higher in the list, is calculated by combining the previous degree of similarity with Yahoo's original rank, using the following weighting scheme:

$$match(user_i, Result_j) = \alpha \cdot sim(user_i, Result_j) + (1 - \alpha) \cdot YahooRank(Result_j)$$

where α gets values between 0 and 1. When α is 0, conceptual rank is not given any weight, and the match is equivalent to the original rank assigned by Yahoo. If α has a value of 1, the search engine ranking is ignored and pure conceptual match is considered. Obviously, the conceptual and search engine-based rankings can be blended in different proportions by varying the value of α.

The final score of each search result - which is a linear combination of the conceptual score and the search engine score - is assigned to the search result as shown in fig (12). Finally, the search results are sorted based on their final scores, so that the ones with higher scores are ranked higher.

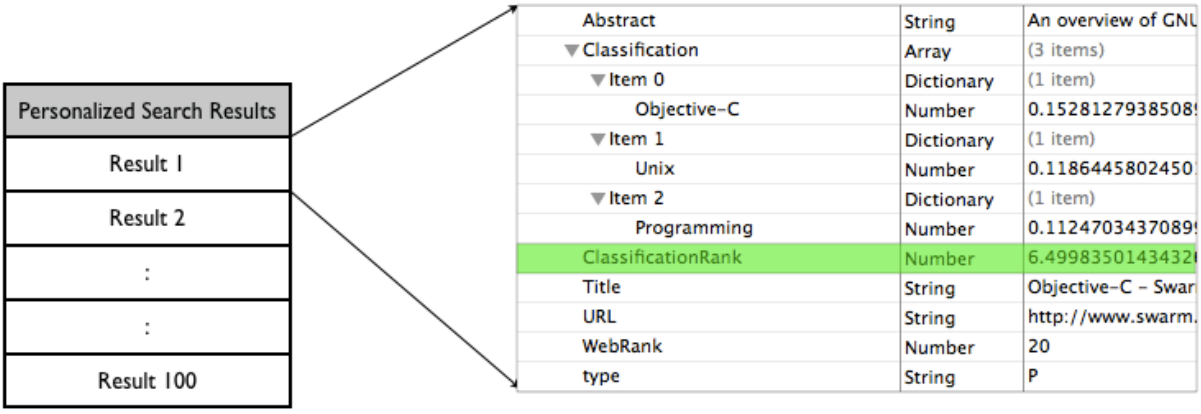


Fig 12 : Post re-ranking, search result details stored on the iPhone

Once the user is presented the re-ranked search results, the system enters into observation mode. Whenever the categories that were assigned to that search result by the classifier. For instance, if categories C1, C2 and C3 were assigned to the search result by the server, C1 being the best match and C2 and C3 being the second and the third best matches, the system would increase the weight of C1 by 3, C2 by 2 and C3 by 1 in the user model. After the user clicks a search result and is viewing a web page, we also monitor the hyperlinks that the user visits from the web page. When the user clicks on a hyperlink, we extract the text from all the paragraph elements of the target webpage. The extracted text is sent to the server for classification. Once the classification results are returned from the server, the top three category matches are updated proportionally in the user model, as described above. In chapter 5, we discuss the evaluation of our personalization system.

Chapter 5

Evaluation

5.1 Text Classifier Training Experiments

Experiment 1: Determining Number of Documents Needed to Train Each Category

We wished to test how the classifier precision varies with the amount of training data. In this experiment, the classifier was trained 30 times starting with 1 training document per category on the first run, and adding 1 training document per category on each subsequent run. For determining the classifier precision, we randomly selected 6 documents per category to be used for testing. There was no overlap between the test documents and the training documents. The classifier precision was calculated as the ratio of the number of test documents that the classifier could accurately classify, to the total number of test documents.

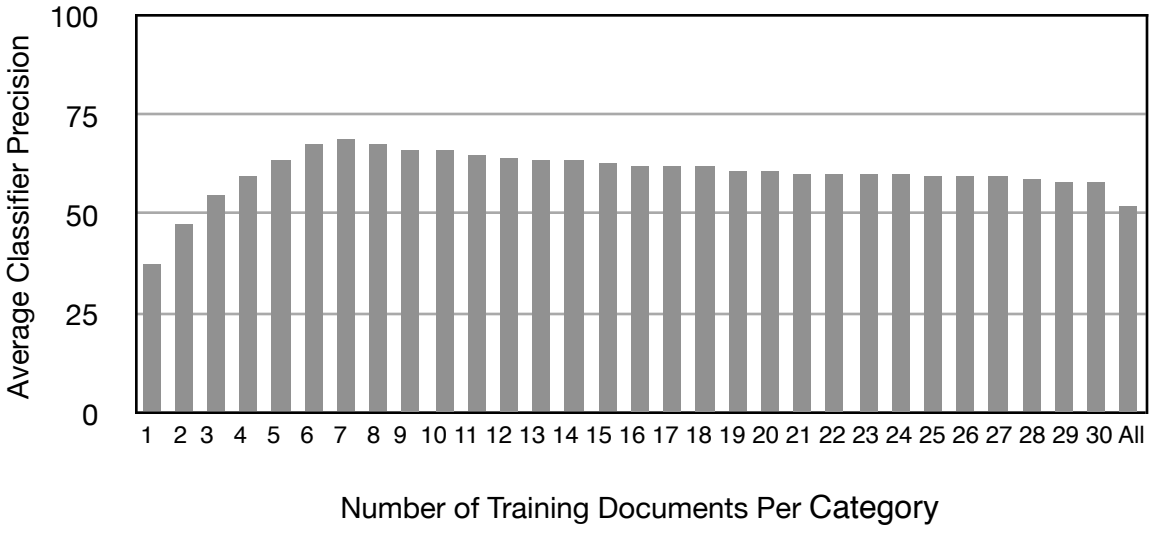


Fig 13: Effect of Number of Training Documents on Classifier Precision

For each classifier, we obtained three precision values by selecting different training and test documents every time. The average precision for a given classifier was calculated by taking the average of all the precisions for that classifier.

As shown in fig (13), the average precision was found to be 37.26% with 1 training document per category. It increased as more training documents were added, till we reached 7 documents per category. For 7 documents, we reached a peak of 69.11% average precision. After that, the classifier precision decreased steadily as more training documents were added until it reached 52.25% when all documents were used for training. Going by the statistics alone, 7 documents per category would be the ideal choice for the training set. But given that the drop in the classifier precision is pretty modest (around 8%) between 7 documents per category and 20 documents per category, we believe it may be a better idea to use somewhere between 7 and 20 training documents per category. This is because the additional training documents will add more features to the classifier. The added features may be useful in discriminating new documents in the future. With this idea in mind, we used 20 training documents per category in our final classifier.

Experiment 2: Determining whether Number of Categories Affects Classifier Precision

This experiment tried to determine whether or not the precision of the classifier is independent of the number of categories between which the classifier must decide. To accomplish this, the classifier was trained on different subsets of the 480 categories. First, 50 categories were chosen at random, the classifier trained on their documents, their test cases submitted to the classifier, and the average classifier precision was determined.

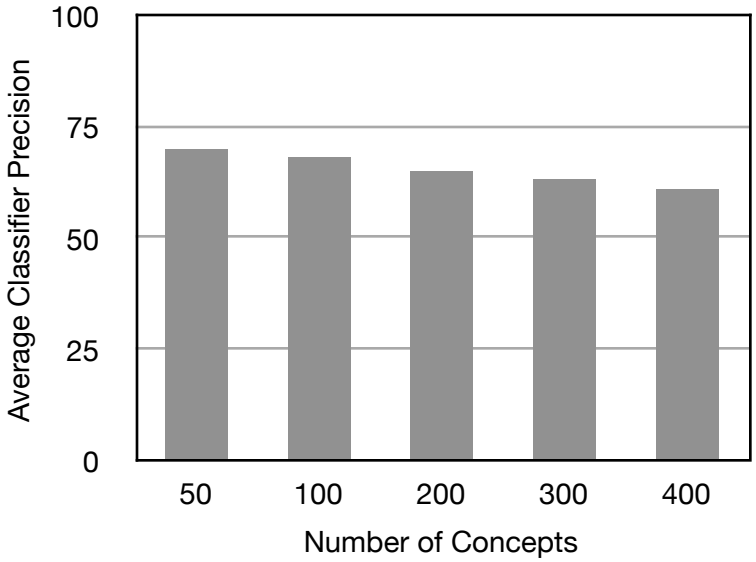


Fig 14: Effect of Number of Categories on Classifier precision

Next, the same procedure was repeated for 100, 200, 300 & 400 randomly selected categories. The average classifier precision was determined in each case. As shown in fig (14), the classifier precision decreases only slightly as the number of categories increases. The precision decreases from 69.89% in case of 50 concepts to 61.25% in case of 400 concepts. This decrease in precision is acceptable especially because it allows us to use more concepts for training the classifier.

Experiment 3: Dependence of Classifier Precision on the Categories Chosen

In this experiment, we wished to determine if the classifier precision varied with the particular set of categories chosen. We trained the classifier on two different sets of 480 categories - Set A) does not contain any regional categories, and Set B) contains 100 regional categories. We calculated average classifier precisions (over three runs) in both cases.

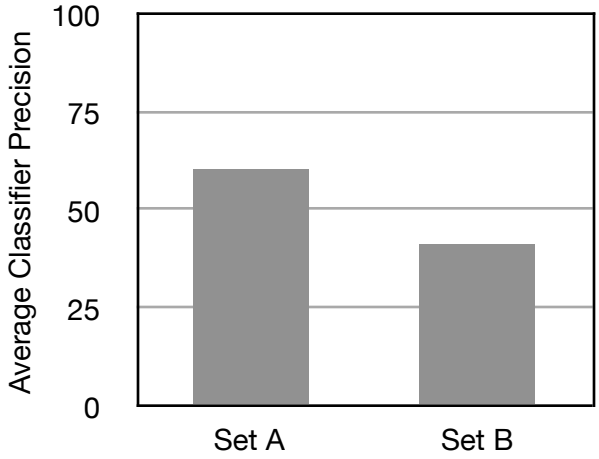


Fig 15: Dependence of Classifier Precision on the Particular Set of Categories

As fig (15) shows, the average precision of the classifier trained on set A was much higher compared to the other. This is because regional categories often contain similar information (e.g., words like ‘county’, ‘district’, ‘state’) and therefore end up confusing the classifier that was trained on set B. We can therefore say that the classifier precision does depend on the particular set of categories on which it is trained and it is very important to pick the right set of categories on which to train the classifier.

Experiment 4: Comparing Performance of classifiers trained by APPROACHES I and II

In this experiment, we wished to test how the classification time varied with the number of features per concept. The first classifier was trained using APPROACH I described in section 4.3, i.e., using all documents under a concept as training documents. The second classifier was trained using APPROACH II, i.e. using 20 training documents per concept. This time, we decided to test the classifiers in a more production like environment. So, we selected 14 sample search queries and for each query we logged the time it took for the classifiers to classify the top 50 Yahoo search results. Appendix C contains the data collected for experimentation, and the results.

Fig (16) compares the time each classifier took to classify the top 50 Yahoo search results. The X axis represents queries and the Y axis represents classification time in milliseconds. It is clear from the graph that APPROACH II consistently outperformed APPROACH I.

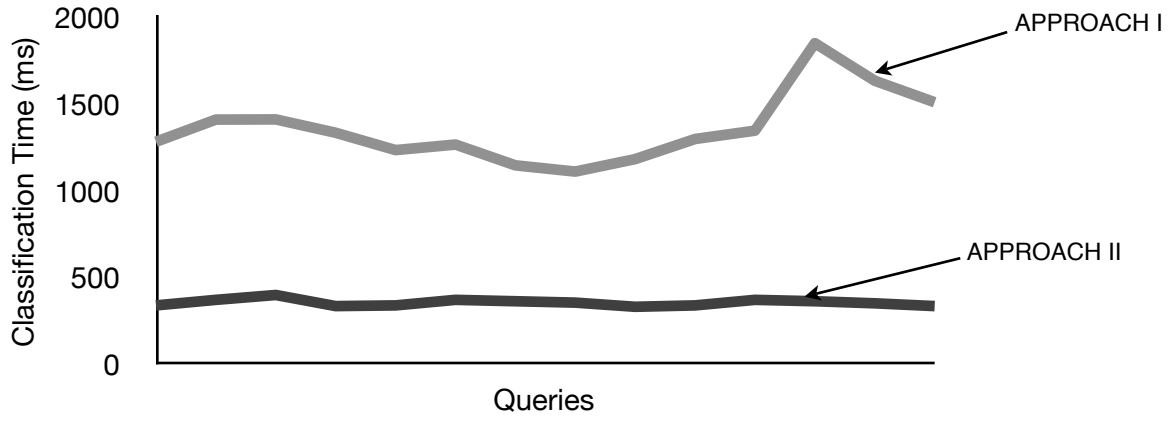


Fig 16: Time taken to classify top 50 Yahoo Search Results

We also selected one search result from the top 50 Yahoo search results for each query. We then recorded the top two classification results of each classifier for the 14 selected web search results. To compare the quality of classification results of the three classifiers, we asked five graduate students from the Computer Science department of the University of Georgia to rate each classification result on a scale of 0 - 1, 0 being incorrect, 0.5 somewhat correct and 1 being correct.

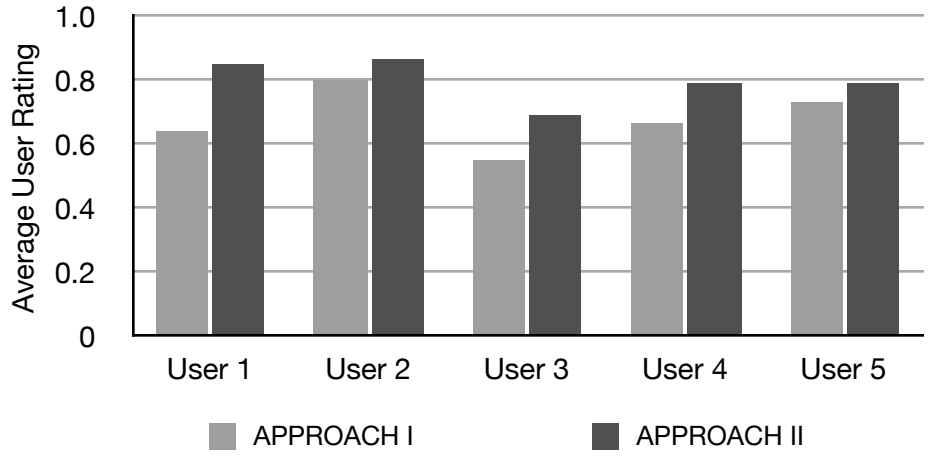


Fig 17: Average User Rating for Classification Results

Fig (17) above compares each user's ratings for the classification results produced by the three classifiers. We notice that all the 5 users considered the classification results of the APPROACH II to be of better quality compared to the first.

5.2 Client Side Experiments

To evaluate the effectiveness of our personalized search results, we built an evaluation version of our client-side system. The evaluation version was designed to be run on the 'iPhone Simulator' software that is part of the iPhone SDK. The iPhone Simulator allows developers to test out their iPhone apps before porting it to the device. Inside the simulator, an app looks and runs exactly like it does on the actual device.

In the evaluation version, we combine the top 10 web search results from Yahoo and the top 10 personalized search results. If there is overlap (e.g., when some of the top 10 personalized search results come from the top 10 Yahoo search results), we add an equal number of personalized and Yahoo search results so that the final count of search results displayed to the user is 20. The search results are shuffled before they are

displayed to the user so as to remove any bias. Upon clicking a search result, the requested web page opens in a new window, and a “Back” button appears that allows navigating back to the search results. We also record the fact that the user considered the selected search result relevant. We communicate this to the user by displaying a small tick mark next to the visited search results. If in fact the user thinks otherwise, he can uncheck the search result and the record for that search result will be removed.

We asked 5 graduate students (3 from Computer Science, 1 from Textile Science and 1 from Bio Technology) from University of Georgia to use the evaluation version of our app over a period of 7 days. In the rest of the discussion, we refer them as User 1, User 2, etc. The users were first given an overview of our system and were explained the experimental setup (describe above). They were asked to use our application for performing web search just as they would normally query a search engine. Before clicking on any search result for a given query, the users were asked to carefully review the title, abstracts and URLs of all the search results and then click on the ones they thought were relevant to them.

Experiment 5: System Generated User Profile vs True User Profile

Given that our primary goal is to learn a model of user interests based on his interaction with search results, and use this model to personalize search ranking, one natural way to evaluate our learning method is to measure the difference between the user’s actual interest vector and the learned interest vector. At the end of the 10 day period of user evaluation, the users were shown the top 20 system predicted user interests were asked to re-order the interests based on what they thought were their true interests.

Appendix D contains system predicted and the true list of user interests for User 1 and User 2. To measure the degree of agreement between the two lists, we calculate normalized Kendall tau distance (see [30], for how normalized Kendall tau distance is calculated) between them. The normalized Kendall tau distance lies in the interval [0, 1], where 0 indicates that the two lists are identical and 1 indicated maximum disagreement.

	Normalized Kendall Tau Distance
User 1	0.19
User 2	0.1
User 3	0.15
User 4	0.27
User 5	0.2

Table 3: Normalized Kendall tau distance between the system predicted interest vector and the true interest vector

Table 3 shows the normalized Kendall Tau distance value for the five users. We note that the value for all users are closer to 0, which indicates agreement between the system generated interest vector and the true user interest vector. We can therefore assert that our learning method does a good job of identifying user interests.

Experiment 6: Comparing User Interaction with Standard and Personalized Results

In this experiment, we wished to determine which search results the users tended to view more often - personalized search results or the standard search results. For each

query, we recorded which search results the user considered relevant. The search results were tagged as ‘P’ if they came from the personalized results and ‘Y’ if they came from standard Yahoo search results and ‘YP’ if they were common to both, the top 10 Yahoo search results and the top 10 personalized results. At the end of the evaluation, we calculated the total number of search results clicked by each user and how many of them were personalized results.

	# Search Results Clicked	# Personalized Results Clicked	# Yahoo Results Clicked	% of Personalized Results Clicked
User 1	171	103	88	60.23%
User 2	229	127	102	55.45%
User 3	226	135	91	59.73%
User 4	160	84	76	52.50%
User 5	174	112	62	64.36%

Table 4: Statistics of the search results clicked

Table 4 compares the percentage of standard and personalized search results clicked by users. It is clear that the users considered the personalized results much more relevant compared to the standard search results. And since the experiment presented search results in an unbiased manner, we can assert that the personalized search results were indeed relevant to user needs and that integrating user interests can help improve the quality of web search.

Chapter 6

Discussion And Future Work

In this chapter, we discuss how our work can be improved in the future. We also discuss additional features that we did not implement in our system due their non-research nature, but which can nonetheless improve the usability of a system like ours in a production environment.

Query auto-complete

Auto-completing queries does a great job of disambiguating ambiguous queries by providing search suggestions. It can also help users save the time needed to write out the complete query. It is indeed a great value add for any type of web search - personalized or traditional. Most of the standard web search engines provide query autocomplete feature in their web search interfaces. The Google, Yahoo and the Inquisitor apps for the iPhone also provide the query autocomplete feature on the iPhone. We did not incorporate this feature in our system because of the logistics involved and because it does not directly relate to personalizing search results.

Integrating a desktop version of our system with the mobile version

Currently, our work focuses only on the mobile platform. But users perform web search from mobile devices as well as their computers. So one way to build better user profiles is to create a desktop personalization system that builds user profile based on the computer and integrate the desktop version of our system with the mobile version. Google does this with their iPhone app and their web search interface.

Document Classifier

Training the document classifier on the right set of categories is extremely important to a personalization system. The document classifier is the component that figures out what a new web page is about and tags the web page with the most relevant concepts from the user model. A document classifier that does not perform its job well will adversely affect the quality of the user profiles generated by the system and by extension, the quality of the personalized results themselves. A document classifier that has been trained on a very broad set of concepts risks introducing ambiguity in the user profile. For example, there is a good chance that such a document classifier will assign a webpage that is about Apple Inc to the concept 'Companies', or a web page that is about Pink Floyd to the concept 'Music Bands', thereby concluding that the user is interested in 'Companies' or 'Music Bands' which does not really say enough about the user. On the contrary, training the classifier on a very fine grained set of concepts will introduce numerous concepts each of which only has a small chance of being assigned to any context. A large number of concepts may actually end up confusing the classifier in cases where two unrelated classes have a similar set of vocabulary. In our research, we therefore carefully picked a set of 480 concepts from the top 4 levels of ODP, using our best judgments and avoiding both the problems discussed above. An improvement to our approach may be to view the final set of concepts as a graph, connecting related concepts by weighted edges. When it is determined that the user is interested in a concept X, we can proportionately increase the weights of the concepts connected to X.

Location and task based profiles

Modern mobile devices are location-aware. Many people also tend to perform web search from some locations more often than others - for example - Home, Work etc. Location based interest profiles that capture user interests at different locations can help deliver personalized results as well. In this case, the user's interests will be distributed across different profiles and the system must combine the different profiles using a weighted scheme before using them for re-ranking. Besides location, additional context information about the user such as the time of the day, weather conditions or the current user activity based on the user's calendar information can be used to provide more granular search result recommendations to the user.

Chapter 7

Conclusion

This research was about personalizing web search on mobile devices. As a case study, we used Apple's iPhone as the client mobile device. Our approach involved building an interest profile on the user's iPhone based on his interaction with web search results and his browsing behavior. Personalization of search results was achieved by re-ranking search results returned by a standard web search engine (Yahoo) based on proximity to the user's interest profile. The ability to recognize user interests in a completely non-invasive way and the accuracy of the personalized results are some of the major advantages of our approach. The average response time of our system for displaying the top 100 personalized search results was found to be less than 2 seconds which is reasonable in a mobile environment. Our experimentation showed that, when presented with an unbiased, randomized list of standard web search results and personalized search results, users viewed personalized results more often than standard web search results. We can therefore assert that search personalization can not only be achieved but can be effective in the mobile environment.

References

1) Jansen, B J, Spink, A, Bateman, J, and Saracevic, T, 1998. Real life information retrieval : A study of user queries on the Web. *ACM SIGIR Forum 32, 1, 5 –17*

2) V. Roto, A. Popescu, A. Koivisto, and E. Vartiainen. Minimap: a web page visualization method for mobile phones. *CHI '06: Proc of the SIGCHI conference on Human Factors in computing systems, pages 35–44, New York, NY, USA, 2006*

3) The Librarian's Internet Index - *http://lii.org*

4) The Internet Public Library - *http://ipl.org*

5) Yahoo Web Directory - *http://dir.yahoo.com*

6) Open Directory Project - *http://dmoz.org*

7) O. Kolesnikov, W. Lee, and R. Lipton. Filtering spam using search engines, 2003.

8) C. Ding, X. He, P. Husbands, H. Zha, and H. D. Simon. Pagerank, hits and a unified framework for link analysis. *In Proceedings of the 25th annual International ACM SIGIR Conference, pages 353–354. ACM Press, 2002.*

9) M. Ester, H.P. Kriegel, and M. Schubert. Accurate and efficient crawling for relevant websites. *In Proceedings of the 30th International VLDB Conference, 2004.*

10) S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *In Proceedings of the 8th Intl. WWW Conference, 1999.*

11) Pitkow, J., Schutze, H., Cass, T., Cooley, R., Turnbull, D., Edmonds, A., Adar, E., And Breuel, T. 2002. *Personalized search. Commun. ACM 45, 9, 50–55*.

12) F. Sebastiani. Machine learning in automated text categorization.

13) ACM Computing Surveys, 34(1):1–47, 2002.

14) Liu, Yu, C., and Mend, W. 2004. Personalized Web search for improving retrieval effectiveness. I*EEE Trans. Knowl. Data Engin. 16, 1, 28–40.*

15) Shun Hattori, Taro Tezuka, and Katsumi Tanaka. Context-aware query refinement for mobile web search. *Symposium on Applications and the Internet Workshops (SAINTW'07).*

16) Leake, D., Scherle, R., Budzik, J., and Hammond, K. (1999). Selecting Task-Relevant Sources for Just-in-Time Retrieval. *In Proceedings of the AAAI-99 Workshop on Intelligent Information Systems. AAAI Press, Menlo Park, CA, 1999*

17) S. T. Dumais, E. Cutrell, E., J. J. Cadiz, G. Jancke, R. Sarin and D. C. Robbins (2003). Stuff I've Seen: A system for personal information retrieval and re- use. *Proceedings of SIGIR 2003.*

18) Critical Mass: The Worldwide State of the Mobile Web, *Nielson Mobile, July 2008*

19) Furnas, G.W., Landauer, T.K., Gomez, L.M., Dumais, S.T.: The vocabulary problem in human-system communication. *Commun. ACM 30(11) (1987) 964–971*

20) http://www.eurekster.com

21) Evgeniy Gabrilovich,Shaul Markovitch:Harnessing the Expertise of 70,000 Human Editors: Knowledge-Based Feature Generation for Text Categorization. *Journal of Machine Learning Research 8 (2007) 2297-2345*

22) http://www.cs.cmu.edu/~mccallum/bow/rainbow/

23) *www.cfigroup.com/news/pressreleases/2009_Smartphone_pressrelease.doc.pdf*

24) http://beckr.org/DBpediaMobile/

25) *http://developer.yahoo.com/search/boss/mashup.html*

26) Mirco Speretta, Personalizing Search Based on User Search Histories, *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (2005) 622 - 628*

27) JSON - *http://www.json.org/*

28) J. M. Madrid, S. Gauch. Incorporating Conceptual Matching in Search

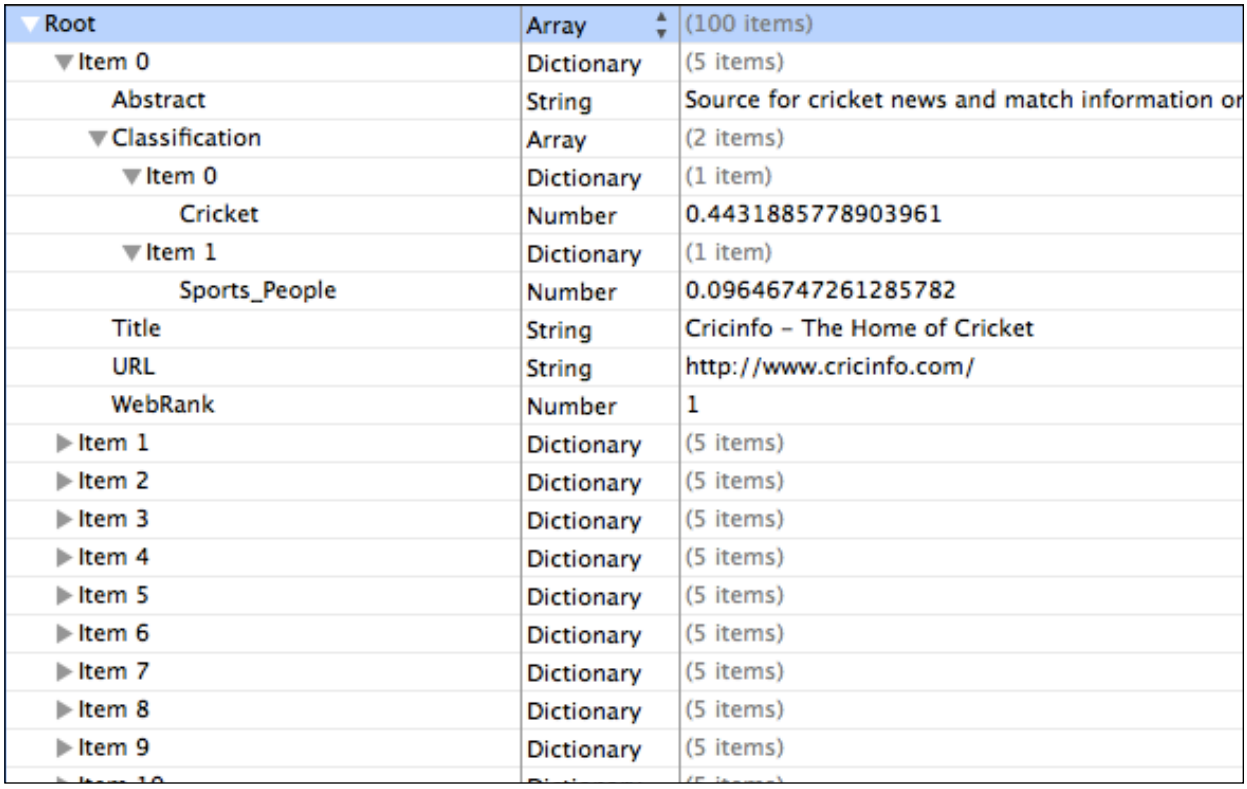
29) Jaime Teevan, Susan T. Dumais and Eric Horvitz. Personalizing Search via Automated Analysis of Interests and Activities. *In Proceedings of the 28th Annual ACM Conference on Research and Development in Information Retrieval (SIGIR '05), Salvador, Brazil, August 2005*

30) Calculating Kendall Tau Distance *http://en.wikipedia.org/wiki/Kendall_tau_distance*

31) Google Alerts - *http://www.google.com/alerts*

Appendix A - Data Structuring

On the user's iPhone, the web search results returned by Yahoo are stored in an array of dictionaries. Each dictionary represents a search result and contains the title, URL, abstract, web rank and classification results for the search result.



Root	Array	(100 items)
▼ Item 0	Dictionary	(5 items)
Abstract	String	Source for cricket news and match information or
▼ Classification	Array	(2 items)
▼ Item 0	Dictionary	(1 item)
Cricket	Number	0.4431885778903961
▼ Item 1	Dictionary	(1 item)
Sports_People	Number	0.09646747261285782
Title	String	Cricinfo – The Home of Cricket
URL	String	http://www.cricinfo.com/
WebRank	Number	1
▶ Item 1	Dictionary	(5 items)
▶ Item 2	Dictionary	(5 items)
▶ Item 3	Dictionary	(5 items)
▶ Item 4	Dictionary	(5 items)
▶ Item 5	Dictionary	(5 items)
▶ Item 6	Dictionary	(5 items)
▶ Item 7	Dictionary	(5 items)
▶ Item 8	Dictionary	(5 items)
▶ Item 9	Dictionary	(5 items)

Fig 1 : Web search results returned by Yahoo are stored in an array of dictionaries on the user's iPhone

Appendix B - Sample Rainbow Classification Results

Query: *cryptography*

Result #: *2*

Search Result URL: *http://cryptography.com/*

Search Result Abstract + Yahoo key terms (sent to Rainbow Classifier)

Cryptography Research DPA smart card Countermeasures Paul Kocher security technology Crypto Resources power analysis media processors Joshua Jaffe piracy data security timing attack Technology Leadership Award ANTI-COUNTERFEITING cryptographic research differential power analysis power consumption encryption cryptology Run by Paul Kocher includes an extensive crypto resources section as well as information on academic research and commercial services

Top 10 categories assigned to the search result by the Rainbow Classifier:

Cryptography	0.3861781061
Computer_Security	0.1622290611
Computer_Science	0.09392657876
Delphi	0.08382996172
Math	0.06551124156
Science_Publications	0.06011695787
Algorithms	0.05566206574
Data_Mining	0.05528485775
Mac_OS	0.05354425311
Visual_Basic	0.05273826793

Top 2 categories sent to the user's iPhone :

Abstract	String	Run by Paul Kocher, includes an extensive crypto resources section as well
▼Classification	Array	(2 items)
▼Item 0	Dictionary	(1 item)
Cryptography	Number	0.3861781060695648
▼Item 1	Dictionary	(1 item)
Computer_Security	Number	0.162229061126709
Title	String	Cryptography Research
URL	String	http://www.cryptography.com/
WebRank	Number	2

Query: *nachos threads*

Search Result #: *30*

Search Result URL: *http://web.engr.oregonstate.edu/~herlock/411/phase1.html*

Search Result Abstract + Yahoo key terms (sent to Rainbow Classifier)

nachos elevator Java threads semaphores priority thread lock kernel TCB bank thread system condition variables KThread project test priority scheduler priority scheduling source files implementation priority elevator riders The only package you will submit is nachos threads so don't add any source files to any other package Your second implementation of condition variables must reside in class nachos threads Condition2

Top 10 categories assigned to the search result by the Rainbow Classifier :

Concurrent_Programming	0.2320144922
Distributed_Computing	0.08116083592
Haskell	0.07864192873
Project_Management_Software	0.06799384952
Backup_Software	0.06668437272
Microsoft_Windows	0.0658243373
BSD	0.06305337697
Information_Technology	0.060878627
Virtual_Reality	0.06055219844
Oracle	0.05692281947

Top 2 categories sent to the user's iPhone :

Abstract	String	The only package you will submit is nachos.threads, so don't add any source files to any othe
▼ Classification	Array	(2 items)
▼ Item 0	Dictionary	(1 item)
Concurrent_Programming	Number	0.2320144921541214
▼ Item 1	Dictionary	(1 item)
Distributed_Computing	Number	0.08116083592176437
Title	String	CS 162 Project Phase 1
URL	String	http://web.engr.oregonstate.edu/~herlock/411/phase1.html
WebRank	Number	30

Query: *software jobs*

Search Result #: *1*

Search Result URL: *http://www.softwarejobs.com/*

Search Result Abstract + Yahoo key terms (sent to Rainbow Classifier)

jobs New Jobs Technology industry Job Search Technology employers Technology dream job New Technology Information Technology Search Process Getting started Get Started inbox Gain Exposure first in line Take two two minutes Post Resume Network member career network Specializing in software and IT search and placement Apply Instantly Be the first in line for your dream job As Featured In A Beyond com Network member recognized as a leading career network

Top 10 categories assigned to the search result by the Rainbow Classifier:

Employment	0.2459572107
Information_Technology	0.1630050838
Software_Engineering	0.1057679877
ERP	0.09317186475
Human_Resources	0.09126808494
Internet_Searching	0.08667631447
Computer_Organizations	0.08586399257
Software_Testing	0.08438787609
Programming	0.08027553558
Project_Management_Software	0.07816051692

Top 2 categories sent to the user's iPhone:

Abstract	String	Specializing in software and IT search and placement. ... Apply Inst
▼ Classification	Array	(2 items)
▼ Item 0	Dictionary	(1 item)
Employment	Number	0.2459572106599808
▼ Item 1	Dictionary	(1 item)
Information_Technology	Number	0.1630050837993622
Title	String	AD&A's Software Jobs
URL	String	http://www.softwarejobs.com/
WebRank	Number	1

Query: *image compression*

Search Result #: *6*

Search Result URL: *http://vision.arc.nasa.gov/publications/mathjournal94.pdf*

Search Result Abstract + Yahoo key terms (sent to Rainbow Classifier)

DCT Mathematica matrix quantization DCT coefficients IDCT 2D frequencies blocks compression array image compression Transpose ShowImage Chop Partition quantization matrix inverse DCT discrete cosine transform InverseFourier show how it is used for image compression We have used these functions in our laboratory to In the JPEG image compression standard each DCT coefficient is quantized using a weight that depends on the frequencies for that coefficient

Top 10 categories assigned to the search result by the Rainbow Classifier:

Image_Processing	0.1339870542
Java	0.1020737141
Data_Formats	0.06163389608
Graphics_Software	0.0487755537
FoxPro	0.04796312749
Algorithms	0.0404240489
Math_Software	0.03966598585
Shopping_Photography	0.03956415132
Software	0.03843202814
Arts_Photography	0.03768092394

Top 2 categories sent to the user's iPhone:

Abstract	String	show how it is used for image compression. We have used these functions in our labor
▼ Classification	Array	(2 items)
▼ Item 0	Dictionary	(1 item)
Image_Processing	Number	0.1339870542287827
▼ Item 1	Dictionary	(1 item)
Java	Number	0.1020737141370773
Title	String	Image Compression Using the Discrete Cosine Transform
URL	String	http://vision.arc.nasa.gov/publications/mathjournal94.pdf
WebRank	Number	6

Query: *christmas movies*

Search Result #: *11*

Search Result URL: *http://www.christmasmovies.us/a-z.html*

Search Result Abstract + Yahoo key terms (sent to Rainbow Classifier)

DVD Christmas featurette Christmas Movies Audio commentary Original theatrical trailer Theatrical trailer The Christmas music video A Christmas Carol double feature Peter Billingsley documentary Interactive trivia Beverly D'Angelo Randy Quaid Johnny Galecki holiday Cartoon Network Christmas Story Welcome to Christmas Movies on DVD Featuring all the classics and even not so classic Christmas movies that have been released on DVD from the black and whites to animated TV specials they're all here

Top 10 categories assigned to the search result by the Rainbow Classifier:

Christmas	0.2503859699
Movies	0.148486197
Holiday_Shopping	0.1053152829
Battlefield_Earth	0.07217542082
Society_Holidays	0.0707449317
Personalized_News	0.06715784967
Mission_Impossible_Series	0.06229489669
Baking_and_Confections	0.05972060561
Batman_Series	0.05791798353
Scream_Trilogy	0.05255771518

Top 2 categories sent to the user's iPhone:

Abstract	String	Welcome to Christmas Movies on DVD. Featuring all the classics (and even
▼ Classification	Array	(2 items)
▼ Item 0	Dictionary	(1 item)
Christmas	Number	0.250385969877243
▼ Item 1	Dictionary	(1 item)
Movies	Number	0.1484861969947815
Title	String	Christmas Movies on DVD: Complete List, A-Z
URL	String	http://www.christmasmovies.us/a-z.html
WebRank	Number	11

Appendix C - Comparing APPROACHES 1 and II for Training a Classifier

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
1	newtons laws	velocity Newton Aristotle acceleration vector Newton's Three Laws of Motion Newton's three Laws Law of Inertia the dynamics Newton's First Law of Motion Newton's Second Law of Motion law accord boat Three Laws of Motion uniform motion state of motion Newton's Third Law of Motion Motion explanation Universe how Newton changed our understanding of the Universe by enumerating his Three Laws of Motion Newton First Law of Motion	1. Law 2. Aerospace and Defense	1284
			1. Quantum Mechanics 2. Mathematical Physics	340
2	Christmas movies	DVD Christmas movies xmas A Christmas Carol Scrooge films classic Christmas christmas not-so-classic TV specials black and white white classic animated TV specials modern classics October 26 Offers a large collection of Christmas movies and TV specials that have been released on DVD sorted by category including old and modern classics "Scrooge" movies comedy and drama	1. Christmas 2. Society Holidays	1412
			1. Movies 2. Christmas	372

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
3	software jobs	jobs technology industry technology Job Search New Technology information technology Dream Job in Just the Quick Step 2 FREE Education Education Information Step 3 Network member career network Syndication Take two two minutes Post Resume Job Seeker Specializing in software and IT search and placement	1. Employment 2. Information Technology	1413
			1. Employment 2. ERP	400
4	tf idf	tf-idf term frequency tf-idf inverse document frequency corpus free encyclopedia information retrieval frequency brown cow dj Gerard Salton scheme Vector Space Model cosine similarity Communications of the ACM statistical measure sum non-relevant documents occurrences Kullback-Leibler divergence The tf-idf weight term frequency inverse document frequency is a weight often used in information retrieval and text mining	1. Information Retrieval 2. Statistics	1336
			1. Information Retrieval 2. Neural Networks	336

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
5	Sodium Chloride	Sodium chloride salt g/100 mL Solubility of NaCl rock salt de-icing table salt Crystal structure free encyclopedia calcium chloride Search compounds Common chemicals atom CaCl2 freezing point mineral common salt lattice point Additives Sodium chloride is also the raw material used to produce chlorine which itself is required for the production of many modern materials including PVC and pesticides	1. Chemicals 2. Inorganic Chemistry	1236
			1. Inorganic Chemistry 2. Chemical Engineering	340
6	rubber soul	The Beatles Rubber Soul Lennon song McCartney instruments recording stereo Harrison Norwegian Wood Allmusic Lead vocals George Martin 06-15 CD mono Lennon and McCartney tracks I'm Looking Through You Beatles song Rubber Soul is the sixth UK studio album and the eleventh US release by the British rock band The Beatles Produced by George Martin and released in December 1965 Rubber Soul had been recorded in just over four weeks to make the Christmas market	1. Beatles 2. Rhythm and Blues	1268
			1. Beatles 2. Pop	372

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
7	sun certified java programmer	certification Sun Certified Java Programmer SCJP Sun Training Exam iPod Sun Certified Training programmers Sun certification exams training products country proficiency exam objectives purchase Solaris Certification United States foundation fundamentals Sun Certified Programmer for the Java Platform Standard Edition 6 CX 310 065 NEW! Upgrade Exam Sun Certified Programmer for the Java Platform Standard Edition 5 0 CX 310 056 Certification Preparation	1. Sun Microsystems 2. Computer Certification	1148
			1. Computer Certification 2. Ada	364
8	wmv to avi	video WMV codecs AVI to WMV WMV to AVI video files DivX XviD MPEG4 Microsoft AVI files Windows Media conversion conversion software MPEG-2 WMV codec Audio Video Interleave Microsoft MPEG4 compression ratios WMV 9 Convert video format AVI to WMV / WMV to AVI Freeware Windows Media Video is one good choice for distributing to a wide people since it has the widest compatibility Windows Media Video WMV	1. Multimedia 2. Video	1112
			1. Data Formats 2. Shareware	356

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
9	blackjack	online games BlackJack games games drinks internet games free internet games Puzzle Board Game AddictingGames classic game BlackJack a Puzzle and Board Game from AddictingGames A classic game of 21 Be sure to get some drinks from the house And if it your house then the drinks really	1. Blackjack 2. Trading Card Games	1184
			1. Gambling 2. Casinos	332
10	kruskal's algorithm	edge arc algorithm minimum spanning tree tree Kruskal's algorithm graph spanning tree forest vertices cycle spanning forest free encyclopedia union final edge edge set Search vertex two trees cycle C Kruskal algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph Animation of Kruskal algorithm Requires Java plugin Create and Solve Mazes by Kruskal and Prim algorithms at cut the knot	1. Algorithms 2. Combinatorics	1300
			1. Algorithms 2. Data Mining	340

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
11	university of georgia	Campaign UGA Georgia University of Georgia The University of Georgia Charities agencies college institution of higher education pumpkin patch UGA researchers H1N1 Eastern Standard Time influenza virus UGA Directory higher education harvest fall family family activity carving Official site of the University of Georgia in Athens Georgia Offers news events a virtual tour and UGA master calendar	1. Georgia 2. Softball	1348
			1. Georgia 2. Colleges and Universities News	372
12	anderson cooper	women Podcast workforce CNN Video Anderson Cooper Anderson Cooper 360 Correspondent reading President Ethics committee the House of Representatives solid place beat Boys Club wage gap brings home Erica Hill Facebook Planet in Peril Anderson Cooper goes beyond the headlines to tell stories from many points of view so you can make up your own mind about the news Weeknights 10 ET Review Anderson cooper journey '360\xb0' Blog Anderson on the new book	1. Science News & Media 2. Society People	1853
			1. News 2. Media	364

#	Query	Search Result Abstract + keyterms	Top to Bottom, top two classification results when using APPROACH I & II respectively	Classification Time (in milliseconds)
13	jsp vs php	jsp jsf Desi Tek frameworks ubuntu NoWhereMan mono product Ubuntu Forums J2EE sun ubuntu linux javaserver faces myfaces Mangiante visual designers languages java.sun google [Archive] Jsp vs php vs asp net Programming Talk Ubuntu Forums > The Ubuntu Forum Community > Other Community Discussions > Development & Programming > Programming Talk > Jsp vs php vs asp net PDA I think you can't really compare jsp and asp net with php	1. Ubuntu 2. ASP	1636
			1. Web Programming 2. Scripting	352
14	cruise deals	Cruise Deals Cruises best cruise deals Cheap Cruises Mon-Thu 24hr Hawaii Cruise Private Balcony roundtrip Last minute Money search here Freestyle cabins Group Cruises Last Minute Cruise Deals Cheap Cruise Deals Carnival Cruises deals on cruises best deals Cruise Deals Last minute specials on cruises Check out deals on cruises now Cruise Deals Our name says it all! Begin your search here for the best cruise deals available!	1. Recreation Travel 2. Mission Impossible Series	1512
			1. Recreation Travel 2. Boating	336

Appendix D: Top 20 concepts from the System Generated User Model, and their true ranks based on user feedback

This appendix contains the true interest profiles and the system predicted user profiles for two users who evaluated our system. For each user, we calculate the normalized Kendall tau distance between the two profiles. Kendall tau distance is a metric that counts the number of pairwise disagreements between two lists, and is a measure of similarity between two different ranked lists on the same items. The normalized Kendall tau distance lies in the interval [0,1], where 0 indicated that the two ranked lists are identical and 1 indicates maximum disagreement.

User 1:

Concept	System Predicted Rank	True Rank
Movies	1	2
Tennis	2	4
Producers	3	13
Yahoo	4	5
Cricket	5	3
Computer Science	6	1
Comedians	7	9
Maps	8	10
Adobe	9	11
Cartoons	10	12
Apple	11	8
Vehicle	12	7
Badminton	13	17
Objective-C	14	14
Actors and Actresses	15	6
Parallel Computing	16	15
Play Groups	17	16
Vegetarian	18	19
Astrophysics	19	18
Office Products	20	20

Normalized Kendall Tau Distance = 0.19

User 2:

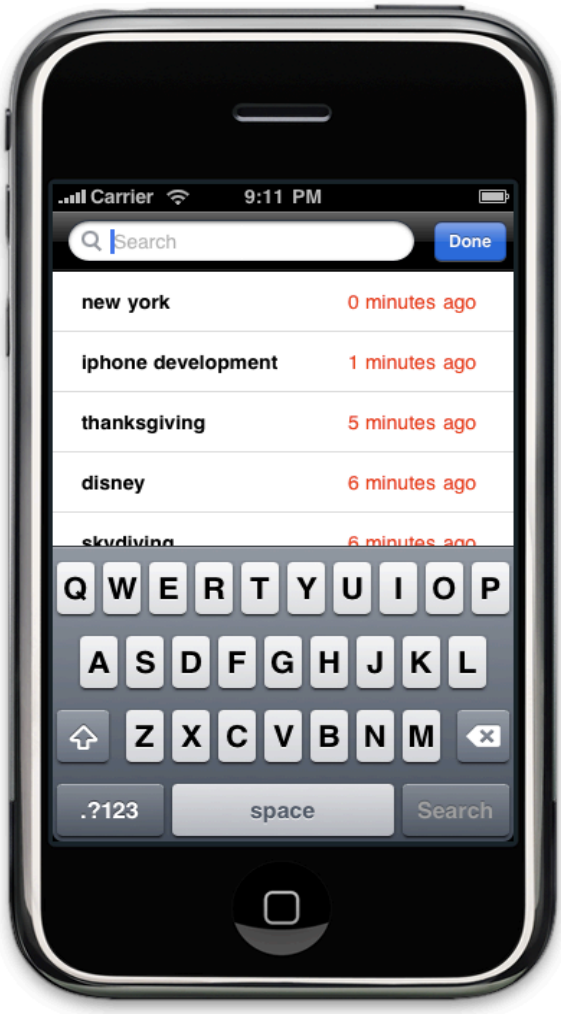
Concept	System Predicted Rank	True Rank
Travel	1	1
Movies	2	2
Vehicles	3	5
Producers	4	4
Actors and Actresses	5	3
Humanities	6	9
Computer Science	7	8
Literature	8	10
Computer Networking	9	11
Automotive	10	6
Music	11	12
E-Commerce	12	13
Cartoons	13	14
Formula One	14	15
Transportation	15	7
Furniture	16	18
Conspiracy	17	16
Classifieds	18	20
Poetry	19	17
Religion	20	19

Normalized Kendall Tau Distance = 0.1

Appendix E: Screenshots

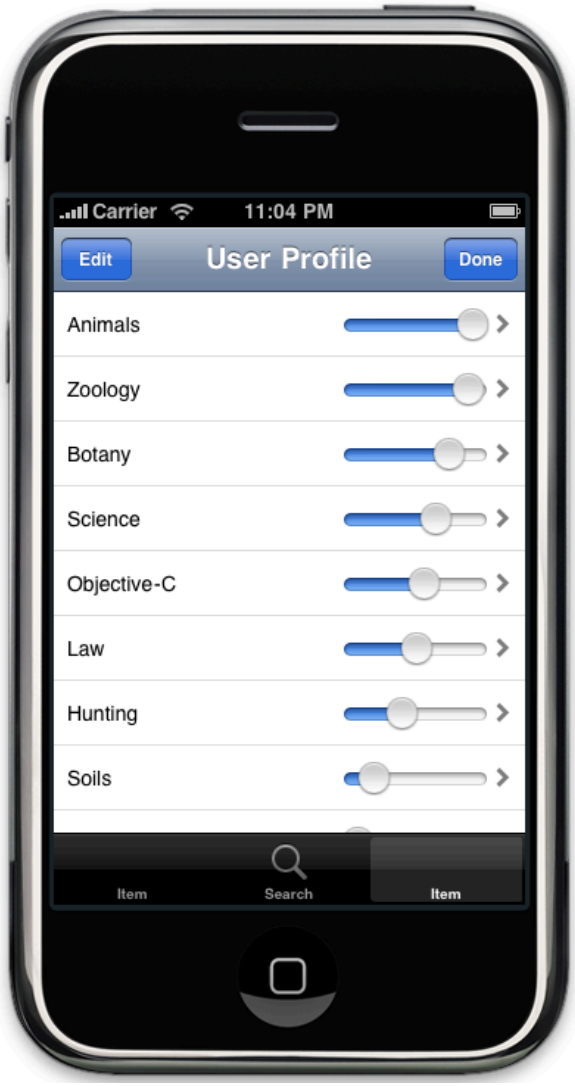
Search Results Display:

Shows how the search results are displayed to the user. Visited results are marked as a visual cue.



Search History:

Shows previous searches along with time when search was made



User Profile:

Automatically updated by the system based on the web pages visited by the user. The user can also control the weights using the sliders provided.