Ontology Based Mobile Retrieval Using Clickthrough Data

Ms. G. Sivagami
Assistant Professor
Department of Computer Science and Engineering
SRM University
Tamil Nadu,India- 603203

Ms. Reshma Rajeev
Department of Computer Science and Engineering
SRM University
Tamil Nadu,India- 603203

Abstract

A search engine is the window to the Internet. The basic premise of a search engine is to provide search results based on query from the user. A major problem in mobile search is that the interactions between the users and search engines are limited by the small form factors of the mobile devices. Hence a practical approach to capturing a user's interests for personalization is to analyze the user's clickthrough data. The user preferences, ranking function optimization and duplicate detection can together be employed for better search results. Unlike other search engines, ranking is based on user preferences rather than number of clicks to retrieve the search results. The proposed method is based on client-server architecture. The intention is to sieve the data so that the preferred and unpreferred results will be correctly classified.

Keywords – personalization, duplicate detection, function optimization

1. Introduction

A dynamic web page is displayed to show the results as well as other relevant advertisements that seem relevant to the query. This forms the basic monetization technique used by many popular search engines. The search engine contains a database that stores these links to the web pages and a framework to decide the sequence/order these results are displayed. With the exponential growth of the web pages and end users demand for optimal search results, there has been a huge push in using data mining techniques to perfect the process of knowledge discovery and understanding the data as well as pre-processing and data preparation.

Table 1 : Search Results for query "Apple"

Links	The list of search results with title, abstract and URLs of Web pages
l1 (clicked)	**Apple Oppurtunities at Apple.Visit other Apple sites… http://www.apple.com/**
l2	Apple-QuickTime-Download Visit the Apple Store online or at retail locations… http://www.apple.com/quicktime/download/
l3	Apple-Fruit Apples have a rounded shape with a depression at the top… http://www.hort.purdue.edu/ext/senior/fruits/apple1.htm/
l4 (clicked)	**Apple .Mac Welcome …member specials throughout the year.See… http://www.mac.com/**
l5	www.apple –history.com A brief history of the company that changed the computing world… http://www.apple-history.com/

A major problem in mobile search is that the interactions between the users and search engines are limited by the small form factors of the mobile devices. As a result, mobile users tend to submit shorter, hence, more ambiguous queries compared to their web search counterparts. In order to return highly relevant results to the users, mobile search

engines must be able to profile the users' interests and personalize the search results according to the users' profiles.

Different users may submit same query but those query may have same or different concept. For example a query apple can have two concepts, apple as a fruit or product related to apple company. The Table 1 gives a list of returned search results for the query "apple". The bold text indicate clicked links. Hence from the clicked links we can conclude that the specific user is interested in the information about the products related to apple company. This difference in concept can be exploited to provide better search results.

2.Related Works

T. Joachims presented an approach to automatically optimize the retrieval quality of search engines using clickthrough data. The goal was to develop a method that utilizes clickthrough data for training, namely the query-log of the search engine in connection with the log of links the users clicked on in the presented ranking. Taking a Support Vector Machine (SVM) approach, this paper presents a method for learning retrieval functions. From a theoretical perspective, this method is shown to be well-founded in a risk minimization framework. Furthermore, it is shown to be feasible even for large sets of queries and features. Theoretically results are verified in a controlled experiment and it shows that the method can effectively adapt the retrieval function of a meta-search engine to a particular group of users.

Another paper published by W. Ng, L. Deng, and D.L. Lee addresses search engine personalization and present a new approach to mining a user's preferences on the search results from clickthrough data and using the discovered preferences to adapt the search engine's ranking function for improving search quality. The approach explains a new preference mining technique called SpyNB, which is based on the practical assumption that the search results clicked on by the user represent the user's preferences, but it does not draw any conclusions about the results that the user did not click on.

E. Agichtein, E. Brill, S. Dumais, and R. Ragno presented a real-world study of modeling the behavior of web search users to predict web search result preferences. Accurate modeling and interpretation of user behavior has important application to ranking click spam detection, web search personalization and other tasks. Here implicit relevance feedback is used reducing the dependence on explicit human judgements.

3.Query Dataset

The search engine must be trained and tested to obtain the desired search results. The dataset used in the training and testing of the search engine is created by Thorsten Joachims of Cornell University. The example set created by Thorsten Joachims consists of 1000 positive and 1000 negative training examples as well as 300 positive and 300 negative training test examples.

Each of the following lines represents one training example and is of the following format:

<line> = <target> <feature>:<value>…..
<feature>:<value> # info

The target value and each of the feature/value pairs are separated by a space character. Feature/value pairs must be ordered by increasing feature number. Features with value zero can be skipped. The string <info> can be used to pass additional information to the kernel (e.g. non feature vector data).

In classification mode, the target value denotes the class of the example. +1 as the target value marks a positive example, -1 a negative example respectively. So, for example, the line

-1 1:0.43 3:0.12 9284:0.2 # abcdef

specifies a negative example for which feature number 1 has the value 0.43, feature number 3 has the value 0.12, feature number 9284 has the value 0.2, and all the other features have value 0. In addition, the string abcdef is stored with the vector, which can serve as a way of providing additional information for user defined kernels.

The file format of the training and test files in ranking mode is the same as for classification mode, with the exception that the lines in the input files have to be sorted by increasing qid. The first lines may contain comments and are ignored if they start with #. Each of the following lines represents one training example and is of the following format:

<line> = <target> qid:<qid> <feature>:<value> <feature>:<value>.....<feature>:<value> # <info>

Features with value zero can be skipped. The target value defines the order of the examples for

each query. Implicitly, the target values are used to generated pairwise preference constraints. A preference constraint is included for all pairs of examples in the example_file, for which the target value differs. The special feature "qid" can be used to restrict the generation of constraints. Two examples are considered for a pairwise preference constraint only if the value of "qid" is the same. For example, given the example_file

3	Qid:1	1:1	2:1	3:0	4:0.2	5:0	#	1A
2	Qid:1	1:0	2:0	3:1	4:0.1	5:1	#	1B
1	Qid:1	1:0	2:1	3:0	4:0.4	5:0	#	1C
1	Qid:1	1:0	2:0	3:1	4:0.3	5:0	#	1D
1	Qid:2	1:0	2:0	3:1	4:0.2	5:0	#	2A
2	Qid:2	1:1	2:0	3:1	4:0.4	5:0	#	2B
1	Qid:2	1:0	2:0	3:1	4:0.1	5:0	#	2C
1	Qid:2	1:0	2:0	3:1	4:0.2	5:0	#	2D

the following set of pairwise constraints is generated (examples are referred to by the info-string after the # character):

1A>1B, 1A>1C, 1A>1D, 1B>1C, 1B>1D, 2B>2A, 2B>2C, 2B>2D.

4.Proposed Work

The proposed work aims at personalizing and further optimizing the search results. When a user submits a query on the client, the query together with the feature vectors containing the user's content and location preferences are forwarded to the server, which in turn obtains the search results from the back-end meta search engine. The content and location concepts are extracted from the search results and organized into ontologies to capture the relationships between the concepts. The server is used to perform ontology extraction for its speed. The feature vectors from the client are then used in RSVM training to obtain a content weight vector and a location weight vector, representing the user interests based on the user's content and location preferences for the reranking. Again, the training process is performed on the server for its speed. The search results are then reranked according to the weight vectors obtained from the RSVM training and then duplication detection is done. Finally, the optimized reranked results and the extracted ontologies for the personalization of future queries are returned to the client.

Ontology update and clickthrough collection at client. The ontologies returned from the server contain the concept space that models the relationships between the concepts extracted from the search results. They are stored in the ontology database on the client. When the user clicks on a search result, the clickthrough data together with the associated content and location concepts are stored in the clickthrough database on the client. The clickthroughs are stored on the clients, so the server does not know the exact set of documents that the user has clicked on. This design allows user privacy to be preserved in certain degree. The returned search results are further optimized by eliminating duplicate links that are directed to similar documents.

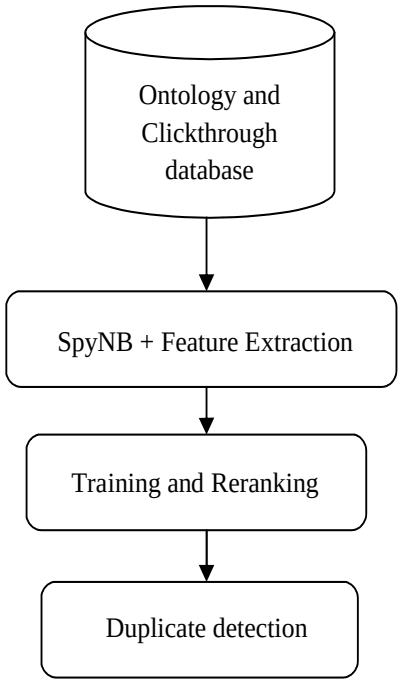

Fig 1 : Proposed Architecture Model Flow

4.1.Ontology and Clickthrough database

Before applying data mining classification algorithm, it is necessary to obtain the clickthrough and ontology data. Previously retrieval was based on training data generated from relevance judgements by experts which is difficult and expensive. The proposed retrieval is based on utilizing clickthrough data. Clickthrough data is the query log of a search engine in connection with log of links the users clicked on in presented ranking. Such clickthrough data is available in abundance and can be recorded at very low cost.

The ontology covers more than what the user actually wants. The concept space for the query "hotel" consists of "map," "reservation," "room

rate,"..., etc. If the user is indeed interested in information about hotel rates and clicks on pages containing "room rate" and "special discount rate" concepts, the captured clickthrough favors the two clicked concepts. Feature vectors containing the concepts "room rate" and "special discount rate" as positive preferences will be created corresponding to the query "hotel". When the query is issued again later, these feature vectors will be transmitted to the server and transformed into a content weight vector to rank the search results according to the user's content preferences. There are several ontology editors that are applications designed to assist in the creation or manipulation of ontology.

4.2.SpyNB and Feature Extraction

The algorithm is based on Naïve Bayes classifier. This method is used to separate the positive and negative training samples created by W. Ng, L. Deng, and D.L. Lee.

It is based on Naïve Bayes classifier.This method is used to separate the positive and negative training samples.The idea behind the procedure is illustrated in Figure. Here P denotes set of all positive links that the user clicked (which may contain links that the user actually wants and donot want) and U denotes unclicked links (which may contain links that the user actually wants and donot want). First, a set of positive examples (links) Pi, are randomly selected from P(set of all positive link) and put in U(unclicked or unlabelled links) to act as "spies". Then, the unlabeled examples in U together with Pi are regarded as negative examples to train the Naive Bayes classifier. The trained classifier is then used to assign posterior probability, to each example in (U ∪ Pi). After that, a threshold, Ts, is determined based on the posterior probabilities assigned to Pi. An unlabeled example in U is selected as a predicted negative example if its probability is less than Ts. The examples in Pi act as "spies", since they are positive and put into U pretending to be negative examples. During the process of prediction, the unknown positive examples in U are assumed to have similar behavior as the spies (Pi). Therefore, the predicted negatives, PNi, can be identified, which is separated from U. As a result, the original U is split into two parts after the training. One is PNi which may still contain some positive items (white region) due to error in the classification. Another is the remaining items in U which may still contain some negative items (black region), also due to error in the classification. Note that Pi returns to P, since it is known to be (sure) positive.

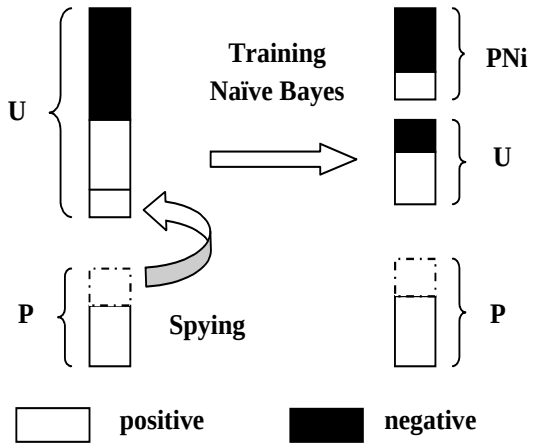

Fig 2 : Spy Technique

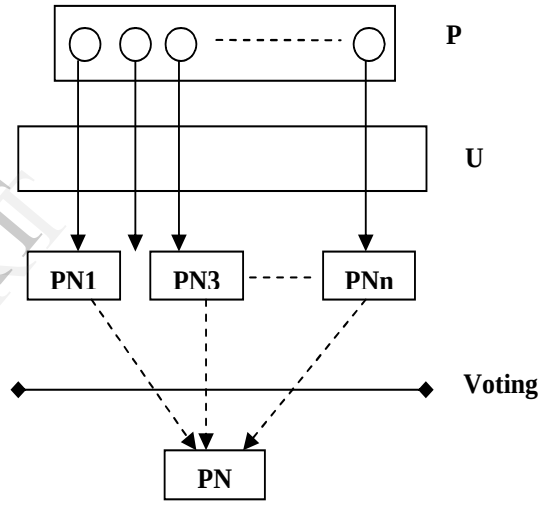

Fig 3 : Voting Method

In the spying technique, the identified PN can be influenced by the selection of spies. As for clickthrough data, there are typically very few positive examples (recall that they are clicked links). To make full use of all the potential spies to reduce the influence. Thus, a voting procedure to strengthen the spying technique further is introduced. The idea of a voting procedure is depicted in Figure and is explained as follows. First of all, the algorithm runs the spying technique n times, where n is the number of positive examples. Each time, a positive example, Pi in P is selected to act as a spy and put into U to train the Naive Bayes classifier. The probability assigned to the spy Pi, can be used as the threshold, Ts, to select a candidate predicted negative set (PNi). That is, any unlabeled example with a smaller probability of being a positive example than the spy is selected into PNi. As a result, n candidate predicted

negative sets PN1, PN2... PNn, are identified. Finally, a voting procedure is used to combine all PNi into the final PN. An unlabeled example is included in the final PN, if and only if it appears in at least a certain number (Tv) of PNi. Tv is called the voting threshold. The voting procedure selects PNs based on the opinions of all spies and thus minimizes the bias of the spy selection.

Choosing an appropriate representation of words in text documents is crucial to obtaining good classification performance. Different representations have been used to maximize the accuracy of machine learning algorithms. The ”Bag of words” representation is widely used to represent text documents. In this representation, a document is considered to be an unordered collection of words whereas the position of words in the document bears no importance. “'Bag of words'” is the simplest representation of textual data. The number of occurrences of each word in the document is represented by term frequency (TF) which is a document specific measure of importance of a term. The collection of documents under consideration is called a corpus. The importance of a term in a document is measured by its weight in the document. A number of term weighting techniques have been proposed in literature. In the vector space model, a document is represented by a document vector whose components are term weights. A document using term frequency as term weights can be represented in vector form as {tf 1,tf 2,tf 3,... ,tf n}, where tf is the term frequency and n is total number of terms in the document.

4.3.Training and Reranking

Training is done by employing Ranking Support Vector Machine(RSVM). Given an independently and identically distributed training sample S of size n containing queries q with their target rankings r*,

(q1, r1*), (q2, r2*),…… (qn, rn*)

the learner L will select a ranking function f from a family of ranking functions F that maximizes the empirical τ on the training sample.

$$\tau s(f) = \frac{1}{n}\sum_{i=1}^{n} \tau(r_{f(q_i)}, r_i^*)$$

To optimize personalization effectiveness, the search results are reranked according to weight vectors obtained for each document returned for a given query.

4.4.Duplicate Detection

A duplication detection algorithm is employed to eliminate those links that direct to the same document. Weighted Component Similarity Summing (WCSS) Classifier is the main algorithm of duplication detection system in identifying duplicates. Inputs to this classifier are the similarity vectors of record pairs from potential duplicates and non-duplicate sets. As we want to develop an unsupervised method there is no training required for this classifier. This classifier tries to find out the duplicates from non-duplicate and potential duplicate datasets. The output from this classifier is a duplicate dataset identified from the potential duplicates and non-duplicate sets.

It is then fed into a support vector machine for classification. The cycle is repeated until no duplicates are found to get the output of SVM as a set of unique records.

5.Experimental Method

In the experimental procedure, users were invited to submit a total of 250 test queries. Each user is assigned five test queries of the same topical category, randomly selected from 15 different categories. In the test phase, a user submits a test query and receives the top 100 search results R from the back-end search engine without any personalization. The user then clicks on any number of results that he/she judges to be relevant to his/her personal interest in much the same way that a standard search engine would have been used.

After the users finished all of the five test queries in the test phase, the training phase begins. The clicked results from the test phase are treated as positive training samples in RSVM training. The clickthrough data, the extracted ontology are employed in RSVM training to obtain the personalized ranking function. After the training phase, the evaluation phase is performed to decide if the personalized ranking function obtained in the training phase can indeed return more relevant results for the user. Each user is asked to provide relevance judgment on all of the top 100 results for each query he/she has tested in the test phase. Documents rated as “Relevant” are considered correct, while those rated as “Irrelevant” are considered incorrect to the user’s needs. The average rank of the relevant documents, for which a lower value indicates better ranking quality.

6.Conclusion

This paper is intended to verify the effectiveness of personalized search results for a given query by different users. The Ranking SVM employed gives much more optimization to search results than SVM. The SpyNB employes a spy as well as voting technique for better classification and the duplicate detection further optimizes the search results.

7.References

1. E. Agichtein, E. Brill, S. Dumais, and R. Ragno, "Learning User Interaction Models for Predicting Web Search Result Preferences," *Proc. Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR)*, 2006

2. K.W.-T. Leung, D.L. Lee, and W.-C. Lee, "Personalized Web Search with Location Preferences," *Proc. IEEE Int'l Conf. Data Mining (ICDE)*, 2010

3. Q. Tan, X. Chai, W. Ng, and D. Lee, "Applying Co-Training to Clickthrough Data for Search Engine Adaptation," *Proc. Int'l Conf. Database Systems for Advanced Applications (DASFAA)*, 2004.

4. T. Joachims, "Optimizing Search Engines Using Clickthrough Data," *Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, 2002.

5. svmlight, http://svmlight.joachims.org/,2012