Learning Query Reformulations for Personalized Web Search Using a Probabilistic Inference Network

Erika Torres-Verdin and Manfred Huber

Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019
{etorresv, huber}@cse.uta.edu

Abstract

The continuous development of the Internet has resulted in an exponential increase in the amount of available pages and made it into one of the prime sources of information A popular way to access this information is by submitting queries to a search engine which retrieves a set of documents. However, most search engines do not consider the specific information needs of the user and retrieve the same results for everyone, potentially resulting in poor results due to the inherent ambiguity in keyword-based search queries. One way to address this is by creating a personalized profile that incorporates the search preferences of the specific user. We present an intelligent system that is capable of learning such a search profile given a set of queries. The search profile is represented with a probabilistic network that incorporates semantic information about the user's use of keywords in the queries. This profile is then used to automatically modify the original queries created by a specific user to improve the degree of relevance between the user's search interests and the retrieved documents. To learn the profile, we create and implement a gradient-based learning algorithm that uses the results of initial user searches to determine query modifications that improve search performance. The proposed intelligent system is a client-side application which operates with an arbitrary keyword-based search engine and which adapts to the preferences of the user as well as to the characteristics of the search engine. We demonstrate the system by learning a search profile that is used to suggest query modifications within a specific domain of interest.

Introduction

In their search for information people often consult different sources and establish data preferences according to their needs. The Internet has rapidly become one of the largest knowledge bases available and a common way to find information is by submitting a query to a search engine, which in turn retrieves a set of related documents. While people with well-defined information needs often have a clear idea as to the kind of knowledge they are looking for, they still find it difficult to express this idea in a few keywords. Moreover, it is frequently a very difficult task to formulate a search query that retrieves precisely the documents that match a particular interest. This paper describes a client-side application that learns a particular user's search profile in the form of a probabilistic network and uses it to suggest custom query modifications based on previous queries and search results. The decision to build the search profile on the client-side was driven largely by the improved degree of privacy and flexibility resulting from a storage and derivation of the profile on the user's computer rather than at the location of the search engine. To learn the profile, the system generates a modified query for every user query, submits both of them to the search engine and allows the user to classify some of the results as either relevant or irrelevant. It then extracts the most representative words from the classified documents and includes them into the network. Using this, and without involving the user, the system then generates a set of new modified queries which it submits to the search engine. It then scores the results by comparing the retrieved URLs with the URLs of the previously classified documents. Using this information, the network is then trained with the best modified query using a gradient-based algorithm. In this way, the network learns and stores the best query modification for every original query.

There have been a number of related research works whose main objective is to help users find interesting web pages. Much of the work on client side personalization has focused on result reranking or on query augmentation. In reranking approaches (Teevan, Dumais, and Horvitz, 2005; Radlinski and Dumais, 2006) search results from the user query are reordered on the client side in order to move the results that best match the interest of the user to the top. Reranking is performed based on a personalized profile derived either only from previously rated search results or from a wide range of information sources such as files on the client computer or the user's navigation history. While these techniques provide excellent privacy and work with existing search engines, the fact that they do not directly influence the search results returned to the client device also poses a number of challenges. In particular, reranking approaches in general require the retrieval of a significant number of search results in order to ensure that they contain a sufficient number of relevant documents.

Moreover, most of them require all of these documents to be downloaded completely in order to facilitate their reordering. To address this problem, server-side techniques have been added that increase the diversity of the returned results in order to improve the likelihood that relevant documents are contained within the top search results (Radlinski and Dumais, 2006). Also, systems have been proposed which use only a simple summary snippet rather than the entire document for reranking. The goals of much of the approach presented in this paper are to derive a personalization that does not require any server-side modifications and that is aimed at minimizing the amount of document downloads by retrieving complete documents only when they are specifically rated by the user. As a result, its goal is to maximize the number of relevant documents among the top ranked query results retrieved by the search engine. To achieve this, it uses query reformulation in order to adjust the user query such that it leads the search engine to return the highest number of relevant results possible within the top ranked documents.
Query expansion techniques (Xu and Croft, 2000) and a wide range of works in relevance feedback (Salton and Bukley, 1990) are focused on adding words to the existing terms of a query. A common characteristic of many of these techniques is that they do not consider the search history of the user but are rather built based on a set of documents. Significant work has focused on building intelligent agents that assist users while they are browsing the Internet; some examples are *Syskill and Webert* (Pazzani and Billsus, 1997), *WebWatcher* (Joachims, Freitag, and Mitchell, 1997), *Letizia* (Lieberman, 1995), *WAWA* (Shavlik et al., 1999). Different research projects have focused on learning a search profile and executing personalized web search such as *Lira* (Babalanovic and Shoham, 1995), *WebMate* (Chen and Sycara, 1998), or *Almathea* (Moukas, 1996). Work in the area of personalized search categories proposes to map the user queries to a small set of categories based on the user profile and general knowledge (Liu, Yu, and Meng, 2002). Although the output of our system is a modification of an original query, the reasoning to suggest query modifications is different from previous query expansion techniques because it is based on the search profile of a particular user and it allows a reformulation of the query that might re-order or potentially remove terms present in the original query of the user. Our system is also different from the agents *Syskill & Webert*, *Letizia*, *Web Watcher* and *WAWA* because is focused on searching and not on browsing. Among these systems, *Syskill & Webert* is the only one that considers representing the user's information need with a query. However, this agent needs to analyze the contents of all the documents retrieved by the search engine in order to evaluate the quality of a modified query. Our system only needs to analyze the URLs of the documents retrieved by the search engine to evaluate the performance of a modified query. This performance is quantified with a metric that permits to rank different queries according to their score. Among personalized agents *Web Mate* is the most similar to our system in the sense that it also considers the different meanings of a word to suggest modified queries. However our representation of the relation between the query terms and their senses is different because we use a probabilistic network instead of a list of words. This network is not only used to characterize the relationships between words and their senses; its main objective is to represent a personalized web search context. Analogous to *Syskill & Webert*, *Web Mate* requires analyzing all the documents retrieved by a search engine to evaluate query performance and does not provide a score to rank different queries. *Web Mate* represents the different search domains with vector of words that do not consider the word sense disambiguation problem. Our system is capable to represent every context of interest with a different probabilistic network that also incorporates the different senses of a word. Our system is different from all the search agents described before in its attempt to capture the meaning of the query words presented by the user and then translate them into a search-engine-specific query that will retrieve a larger number of relevant documents than the original query. In terms of the learning algorithm, the main contribution is its ability to learn the search profile represented in the network using only the results from the automatically issued modified queries in terms of the retrieved URLs previously classified as relevant and irrelevant. Also, this algorithm does not follow a traditional supervised learning approach since the network does not actually output a search query but rather ranks and rates words as to their efficacy to retrieve relevant documents when used in a particular search engine.
The remainder of the paper first describes the construction of the network for a personalized search profile and the procedure to infer modified queries from it. The next section then describes the algorithm used to learn a profile before the approach is illustrated by learning search profiles for different information needs. The system is then compared with other approaches and the last section finally concludes and gives directions for future work.

Building The Search Profile

The search profile of a particular user is represented with a probabilistic network. Every time the user presents a new query to the system the structure of the network is changed to include the words in the query and their different meanings as identified by an electronic dictionary. In addition, the most representative words of the documents classified by the user as relevant and their respective meanings are added. In order to select the most representative features every original query and its associated relevant and irrelevant documents are stored in a small database. The user does no need to classify all of the retrieved documents, but it is important to notice that the larger number of classified documents, the easier it will be for the system to infer the user's intention. The contents of these documents are analyzed to remove HTML tags and stop words to select the most significant words. The first selection criterion requires that the ratio of relevant to irrelevant documents is better that the original ratio. The second criterion selects words whose entropy loss value (Rosenfeld, 1996) is above a particular threshold defined as a percentage of the highest entropy loss value. In order to break ties, words with identical entropy loss values are grouped and the ones that together best cover the relevant documents are chosen.

The Network Topology

The structure of the probabilistic network consists of three different types of nodes arranged in three levels (Figure 1).

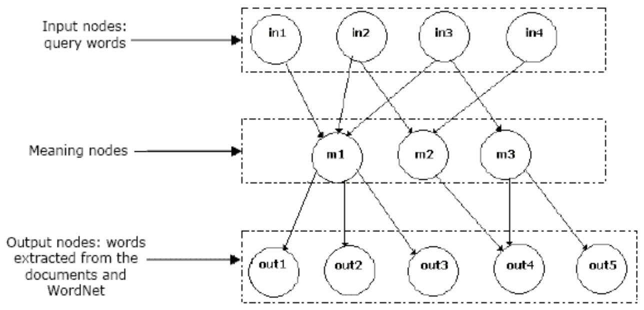


Figure 1. Network Topology.

The first level of the network is composed of the input nodes which represent the words in the original user query. The second level contains the meaning nodes that represent the senses of the query words and of the terms extracted from the relevant documents. These senses are obtained from *WordNet* (Fellbaum, 1998) and are selected according to their associated frequency count (Fellbaum, 1998). The purpose of meaning nodes is to "understand" to which of the particular senses of a word the user is referring in his/her query. The third level of the network is composed of the output nodes that represent the keywords in the modified query suggested by the network. An important difference between the output nodes and the meaning nodes is that the values of the former represent utilities and not strictly beliefs. The value of an output node indicates the utility of a word in expressing a particular meaning to the search engine and in successfully retrieving relevant documents. The output nodes with a utility higher than a particular threshold set as a percentage of the highest utility are used to create the output query associated with an input query. The words in the output query are ordered in descending order of their utility.

Network Connections

Every time a new keyword appears in an original query the topology of the network changes by adding new nodes and connections. A keyword in an original query is symbolized by an input node and a meaning node for each of its senses as identified by *WordNet* is introduced. The input node representing the keyword is connected to all its meaning nodes. In addition, it is connected to all meaning nodes that symbolize the senses of keywords that appear in the same query. For example, with the query *"apple producer"*, the input node *"apple"* is connected to all nodes representing its senses and also to all nodes representing the senses of *"producer"*. For each of the meaning nodes, output nodes representing all the words associated with this meaning according to *WordNet* are created and connected to it. Some of the output nodes are not related to the senses of the keywords in a query since they represent the words that have been extracted from the relevant documents. For these nodes *WordNet* is consulted to find and link the set of corresponding meaning nodes. Then, existing input nodes are connected to this set of meaning nodes by linking all the input nodes to the additional meaning nodes

Initialization of the Values of the Nodes

Once the network topology is determined, the values of the CPT (Conditional Probability Table) entries of each node have to be initialized. For input nodes this value indicates the presence of the keywords in the input query. For meaning and output nodes the word frequencies provided by *WordNet* are used to set the initial CPT values.

Initialization of the Input Nodes. The input nodes are set to *T* (TRUE) or *F* (FALSE) according to the words that appear in the input query. For example, in a network with input nodes *"A"*, *"B"* and *"C"*, the first two are set to *T* and the third to *F* if the user query *"AB"* is presented.

Initialization of the Meaning Nodes. CPT's for the meaning nodes are initialized according to the relative frequency that WordNet assigns to each of the senses.

a) Meaning nodes with one parent: Suppose that there are two input nodes, *"apple"* and *"producer"*, respectively. According to *WordNet* the word *"apple"* has one meaning and the word *"producer"* has three meanings. The ratios of the frequency of each sense of *"apple"* and *"producer"* are summarized in Table 1 and transformed into probabilities.

P (apple1=T | apple=T) = 2/2
P(producer1=T | producer=T) =9/16
P (producer2=T | producer=T) =5/16
P(producer3=T | producer=T)=2/16

Word	Sense	Frequency Ratio
Apple	Apple1	2/2
Producer	Producer1	9/16
	Producer2	5/16
	Producer3	2/16

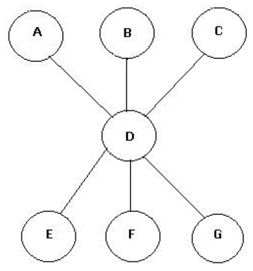
Table 1. Frequency ratios of the Meaning Nodes

b) Meaning nodes with two or more parents: If two or more input nodes represent words that have a common meaning according to *WordNet* then the value of the word with the highest frequency is used to initialize the meaning node. When the input nodes do not share a common meaning, we only consider the words for which *WordNet* gives an answer. For example, in the query *"apple* *producer"*, the input nodes *"apple"* and *"producer"* are linked to the meaning nodes *"apple1"*, *"producer1"*, *"producer2"* and *"producer3"*. *WordNet* does not provide a sense that links the word *"apple"* with the senses of *"producer"* or *"producer"* with the senses of *"apple"*. The meaning nodes *"producer1"*, *"producer2"* and *"producer3"* are thus initialized considering only the information for the word *"producer"*. The same applies for the meaning node *"apple1"*. When *WordNet* does not provide a frequency count for a meaning, as in the case of brand names, the probabilities are initialized to a low value such as *0.01*.

Initialization of the Output Nodes. Output nodes are initialized in a similar way as meaning nodes. However it is important to remember that the values of output nodes represent utilities, normalized to values between *0* and *1*. In the case of output nodes with one parent, *WordNet* is consulted to determine how frequently the corresponding words have the sense assigned to them trough the meaning nodes. These frequencies are translated into probabilities. For an output node with more than one parent the utility value is calculated as a combination of all possible states (*T/F*) of the parents using the Noisy-OR gate (Pearl, 1988).

Prediction of a Query Modification Using Previously Learned Queries

The values of the CPTs were initialized using only the information provided by *WordNet*. The user's queries present new evidence to the network that is used to update the CPTs. Once the network has learned the target output for one or more queries, these can be used to predict the output of a new query. The prediction is not straightforward because only the CPT entries of meaning nodes corresponding to combinations of input nodes used in the given query are updated while all others remain unchanged. The case of the output nodes is different because they do not directly depend on the input nodes, and the learning algorithm (presented in the Learning a Search Profile Section) updates all the values of their CPT entries. Suppose that the small network in Figure 2 has learned the



A	B	C	P(D|ABC) T	F
T	T	T	Not Learned	Not Learned
T	**T**	**F**	**Learned**	**Learned**
T	F	T	Not Learned	Not Learned
T	F	F	Not Learned	Not Learned
F	**T**	**T**	**Learned**	**Learned**
F	T	F	Not Learned	Not Learned
F	F	T	Not Learned	Not Learned

Figure 2. Probabilistic network and CPT for node D

best query modification for the original queries $Q_1=AB$ and $Q_2=BC$ and it has to predict the best modification for the query $Q_3=AC$. In this situation, the CPT entry of node *D* for {*A=T, B=F, C=T*} has to be predicted using the CPT entries corresponding to {*A=T, B=T, C=F*} and {*A=F, B=T, C=T*}. We can obtain Q_3 from Q_2 and Q_1 using the following equations:

$$\frac{P(D=T \mid Q_3)}{P(D=T \mid Q_1)} = \frac{P(D=T \mid A=T, B=F, C=T)}{P(D=T \mid A=T, B=T, C=F)} \quad (1)$$

$$\frac{P(D=T \mid Q_3)}{P(D=T \mid Q_2)} = \frac{P(D=T \mid A=T, B=F, C=T)}{P(D=T \mid A=F, B=T, C=T)} \quad (2)$$

Assuming that all input queries are equally likely and that all input nodes are independent given the meaning node *D*, Bayes Law can be applied to obtain:

$$z_1 = P(D=T|Q_3) \cong P(D=T \mid Q_1) \cdot \frac{P(B=F \mid D=T)}{P(B=T \mid D=T)} \cdot \frac{P(C=T \mid D=T)}{P(C=F \mid D=T)} \quad (3)$$

$$z_2 = P(D=T|Q_3) \cong P(D=T \mid Q_2) \cdot \frac{P(A=T \mid D=T)}{P(A=F \mid D=T)} \cdot \frac{P(B=F \mid D=T)}{P(B=T \mid D=T)} \quad (4)$$

Using $P(D=T|Q_1)$ and $P(D=T|Q_2)$, the unknown ratios of Equations 3 and 4 can be found where ratios that have never been observed are assumed to be *1*. $x = P(D=T|Q_3)$ can then be estimated from z_1 and z_2 in a way similar to estimating a constant quantity *x* from *n* noisy measurements z_i $(i=1,..,n)$ of *x*. To estimate $x=P(D=T|Q_3)$ from z_1 and z_2, the following formula (Kalman Filter) is used:

$$\hat{x} = \left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\right) \cdot z_1 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right) \cdot z_2$$

where $\hat{x}$ is the estimate of *x* and ${\sigma_1}^2$ and ${\sigma_2}^2$ represent the errors (variance) in z_1 and z_2, respectively. The following illustrates the estimation of the variances using ${\sigma_1}^2$ as an example. Let the ratios from Equations 3 and 4, $\frac{P(B=F \mid D=T)}{P(B=T \mid D=T)}$ and $\frac{P(C=T \mid D=T)}{P(C=F \mid D=T)}$, be *J* and *K*, respectively. In the absence of other information, *J* and *K* can be assumed to be the expected values of two independent random variables, *J'* and *K'*, that represent ratios of probability values for different combinations of input nodes. If it is also assumed that the errors in these predictions are small compared to their expected values, i.e. $\sigma_{J'} << J$ and $\sigma_{K'} << K$, then the variance of *J'* and *K'* can be estimated as $\sigma_{J'}^2 = \beta \cdot J^2$ and $\sigma_{K'}^2 = \beta \cdot K^2$, where β is a small constant. The standard deviation of z_1 is then $\sigma_{z_1}^2 = P(D=T \mid Q_1) \cdot \sigma_{J'K'}^2$ and the variance $\sigma_{J'K'}^2$ is $\sigma_{J'K'}^2 = \sigma_{J'}^2 \cdot \sigma_{K'}^2 + \sigma_{J'}^2 \cdot E[J'] + \sigma_{K'}^2 \cdot E[K']$ where *E[J']* and *E[K']* are the expected values of *J'* and *K'* respectively. In this case *E[J']=J* and *E[K']=K*. Although Equations 3 and 4 contain only two ratios, the approach can also be applied in cases with more rations and a new query can thus be predicted given any number of learned queries. As the network learns more queries, the value estimates for the CPT entries should improve, leading to better query modifications over time.

Learning the Search Profile

The system presented here is an intelligent agent that learns a search profile from queries created by a particular user and the classification that he/she assigns to some of the retrieved documents. For every original query the user does not actually provide information as to what the best query reformulation would be but rather only gives information about the quality of a subset of the retrieved documents. The tasks of the learning algorithm have therefore to be: (i) discover which query modifications would be the best; (ii) determine if all the best modified queries can be achieved within the network, i.e. establish what meaning nodes would best represent the intention in the original queries and how these nodes can be best represented in the output nodes to form modified queries that successfully retrieve relevant documents when they are actually submitted to a particular search engine.

The procedure to learn a search profile is as follows. For every original query the system internally creates a set of query modifications using the different meanings of the words in the query and the terms extracted from the classified documents. Every query modification is submitted to the search engine and the URL's of the retrieved documents are compared with those of the classified documents to calculate an expectation of relevant documents. The learning process then consists of updating the CPTs in the network such that given the words in the user's query (input nodes) the system produces the best modified query (output nodes). The CPTs of the network are updated using a gradient-based algorithm using a set of training examples indicating the direction of change of the CPT values. The training data consists of a set of original queries, their modifications, and a quality score. To train the network, a mapping mechanism is essential that converts desired output queries to utility values for the output nodes. The main steps of the learning process are:

a) Set up a measure to score the quality of the modified queries derived from the user's original queries.

b) Map the target queries to values of the output nodes of the network. Use these values to construct a training set.

c) Use a gradient-based learning algorithm to update the CPTs of the network with the training examples from b).

Query performance measure

Let S be the sample space of all combinations of relevant and irrelevant documents retrieved by a search engine in response to a query. Each element of S is a set of documents, denoted by s_r, where r is the number of relevant documents: $S=\{s_0,s_1,\dots,s_N\}$. Given no further information, the prior probability of each outcome set, $P(s_r)$, is given by:

$$P(s_r)=\binom{M}{r}p^r q^{M-r}=\frac{M!}{r!(M-r)!}p^r q^{M-r}$$

where M and $M\text{-}r$ are the total number of documents and the number of irrelevant documents respectively. The constants p and q are the probabilities of relevant and irrelevant documents, respectively. We use $P(s_r)$ to estimate the prior probability of every set of documents to be retrieved by a modified query. The values of p and q are obtained from the set of documents retrieved by the original query. Suppose that we submit the modified query to the search engine and the relevance of some of the retrieved URLs is already known because the associated documents have been classified previously. Although, the classification of the remaining URLs is unknown, we can estimate the performance of the modified query by considering the classified URLs as k randomly picked documents from the total set of URLs retrieved by the modified query. Given that among the k selected documents x are relevant and y are irrelevant, the posterior probability of s_r is:

$$P(s_r|x,\quad y\quad are\_observed\ )=\frac{P(x,\quad y\quad are\_observed\ |s_r)\cdot P(s_r)}{P(x,\quad y\quad are\_observed\ )}$$

The expectation, Z, of the number of relevant documents retrieved by a modified query is calculated as:

$$E(Z)=\sum_{r=0}^{N} r\cdot P(s_r|x,\quad y\quad are\_observed)$$

Using this expectation formula we can score queries and the best modified query will be the one whose expectation $E(Z)$ of obtaining relevant documents is the largest.

Learning the best modification

The creation of training examples from an original query and its best modification (target query) proceeds as follows:

1) Set the input nodes that represent the words in the input query to True and the rest of the input nodes to False.

2) Obtain the current output query from the network.

3) Compare the output query with the target query

4) If the output query and the target query are the same the algorithm is completed, otherwise update the values of the output nodes until the network produces the target query.

The output query produced by the network is composed of the output nodes whose value is above the threshold $threshold = percentage \times highest\_output\_value$. The words in the output query are set in decreasing order of their output node values. The target query is mapped to a set of output node values calculated as the closest value for each output node O_i in the target query that would put it in the correct position with respect to the other words in the target query. The difference between Oi and the rest of the output nodes in the target query is calculated individually using the formulas in Figure 3.

```
A and B are output nodes that symbolize words that must appear in the
target query
Value (A) is the utility value of node A
Value (B) is the utility value of node B
Case 1:
Target Output: Value (A) > Value (B)
Current Output: Value (A) < Value (B)
    If (Value (B) ≥ Threshold)
        Value (A) = Value (A) + α ((Value (B) + ε) – Value (A))
    Else if (Value (B) < Threshold)
        Value (A) = Value (A) + ((Threshold + ε) – Value (A))
Case 2:
Target Output: Value (A) < Value (B)
Current Output: Value (A) > Value (B)
    If (Value (B) ≥ Threshold and Value (A) ≥ Threshold)
        Value (A) = Value (A) + α ((Value (B) – ε) – Value (A))
    Else if (Value (B) < Threshold and Value (A) ≥ Threshold)
        Value (A) = Value (A) + α ((Threshold – ε) – Value (A))
    Else if (Value (A) < Threshold)
        Value (A) = Value (A) + ((Threshold + ε) – Value (A))
Case 3:
Target Output: Value (A) < Value (B)
Current Output: Value (A) < Value (B)
Since the current output is equal to the target output Value(A) is not updated.
Case 4:
C is an output node that is NOT part of the target output, but it appears in the
current output.
Target Output: Value (C) < Threshold
```

Figure 3. Cases used to update the output nodes

The cases in Figure 3 follow the same general formula $Value(O_i)=Value(O_i)+\alpha[(targetValue\pm\varepsilon)-Value(O_i)]$ where α is the learning rate and ε is a margin used to make the output node values more stable.

Each individual difference between an output node O_i and another output node that must appear in the target query is stored in a vector of differences named *Delta*: $Delta=\{\Delta_1,\Delta_2,\Delta_3,\ldots,\Delta_N\}$. Finally, the average of all Δ_j's is used to update the output nodes:

$value(O_i)=value(O_i)+\alpha(Sum(Delta)/size\_of\_Delta)$, where *size_of_Delta* is the number of Δ_j's in *Delta*. Every time the value of an output node is changed, the CPTs of the network are updated using a gradient-based algorithm (presented in the next section).

Learning the profile

Every time a training example is presented to the network, the output nodes must be updated to produce the desired output query given an input query. The error in the values of the output nodes is defined as: *Error=Target Value – Current Value*. To reduce this error we apply gradient-descent on the square error of every output node.

A training example for the network has the form:

Input: $I_1=T$, $I_2=T$, $I_3=F$

Output: $O_1=V_1$, $O_2=V_2$, $O_3=V_3$…..$O_N=V_N$.

where O_i is an output node and V_i its desired value.

Let O_i be an output node of the network, Π_i the set of all parent nodes of O_i and Π_{ij} the jth assignment of the states (true or false) of Π_i. We define $P(O_i=T)$ as:

$$P(O_i=T)=\sum_{j=1}^{2^N}P(O_i=T|\Pi_{ij})\cdot P(\Pi_{ij})$$

where N is the number of parents of O_i and 2^N is the total number of possible assignments of states to the parents of O_i. The problem to be solved is to modify $P(O_i=T)$ according to the training examples. We can see $P(O_i=T)$ as a function with parameters $P(O_i=T|\Pi_{ij})$ and constants $P(\Pi_{ij})$. $P(O_i=T)$ can be updated by following the gradient:

$$\frac{\partial P(O_i=T)}{\partial P(O_i=T|\Pi_{ij})}=P(\Pi_{ij})$$

The result is multiplied by ΔO_i which indicates the magnitude by which the value should change. We define ΔO_i as $P(O_i=T)_{target}-P(O_i=T)_{current}$ and update $P(O_i|\Pi_{ij})$ using:

$$P(O_i=T|\Pi_{ij})=P(O_i=T|\Pi_{ij})+\alpha\cdot\left[\Delta O_i\cdot\frac{\partial P(O_i)}{\partial P(O_i=T|\Pi_{ij})}\right]$$
$$=P(O_i=T|\Pi_{ij})+\alpha\cdot[\Delta O_i\cdot P(\Pi_{ij})]$$

where α is the learning rate.

Each conditional probability is then normalized such that $P(O_i=T|\Pi_{ij})+P(O_i=F|\Pi_{ij})=1.0$ and $P(O_i=T|\Pi_{ij})\in[0,1]$.

After updating the output nodes, the CPT entries of the meaning nodes have to be updated. Let M_k be a meaning node that is a parent of the output node O_i for which we calculated $P(O_i=T)$. We derivate $P(O_i)$ with respect to $P(M_k=T)$ for every child node O_i of M_k and then calculate the average of these L partial derivatives:

$$\frac{\sum_{i=1}^{L}\frac{\partial P(O_i=T)}{\partial P(M_k=T)}}{L}$$

Each of these partial derivatives is again multiplied by ΔO_i of every child node O_i of M_k:

$$\Delta M_k=\frac{\sum_{i=1}^{L}\Delta O_i\cdot\frac{\partial P(O_i=T)}{\partial P(M_k=T)}}{L}$$

The value ΔM_k is calculated for every meaning node, M_k that is a parent of the output node whose value needs to be updated. The last part of the algorithm consists of updating the conditional probabilities of the meaning nodes considering the relationship with the input nodes. We define $P(M_k=T)$ in a similar manner as $P(O_i=T)$:

$$P(M_k=T)=\sum_{j=1}^{2^R}P(M_k=T|\Pi_{kj})\cdot P(\Pi_{kj})$$

where R is the number of parents, Π_k, of meaning node M_k and $2R$ is the total number of possible assignments, Π_{kj}, of states of the parents of M_k. Now the partial derivatives of the previous equation with respect to $P(M_k=T|\Pi_{kj})$ is:

$$\frac{\partial P(M_k=T)}{\partial P(M_k=T|\Pi_{kj})}=P(\Pi_{kj})$$

Finally we update $P(M_k=T|\Pi_{kj})$:

User query	Modified query	Modified query score Rel/Total	Original query ratio Rel/Total	Modified query ratio Rel/Total
Apple consumer	Apple consumer information	7.7/15	5/15	13/15
Apple information	Apple information apples	9.8/18	8/18	15/18
Apple producer	Apple producer growers	9.2/19	8/19	19/19
Apple virus	Apple virus delicious	10/18	7/18	15/17

Table 2 Results for the search profile of User 1

User query	Modified query	Modified query score Rel/Total	Original query ratio Rel/Total	Modified query ratio Rel/Total
Apple consumer	Apple consumer information Mac	9.6/17	8/17	17/17
Apple information	Apple information computer	10.9/18	10/18	18/18
Apple producer	Apple producer Mac	9.4/15	8/15	14/15
Apple virus	Apple virus OS	11.6/17	10/17	17/17

Table 3 Results for the search profile of User 2

$$P\left(M_k = T \middle| \Pi_{kj}\right) = P\left(M_k = T \middle| \Pi_{kj}\right) + \alpha \cdot \left[\Delta M_k \cdot \frac{\partial P\left(M_k = T\right)}{\partial P\left(M_k \middle| \Pi_{kj}\right)} \right]$$

Again, each conditional probability of the meaning nodes must be normalized so that $P(M_k=T \mid \Pi_{kj}) \in [0,1]$ and the entries corresponding to a particular conditioning case Π_{kj} must sum to 1. Given these equations we can now learn and store the best modified queries without user involvement.

Experiments

We have designed two experiments to evaluate different characteristics of the system. In the first experiment, search profiles for two persons with different search preferences were created using the same input queries. The second experiment evaluates the quality of the network's query modification produced for a novel input query. The network creates this output query based on the previously learned queries. In both experiments, the threshold for words extracted from the relevant documents was set to *0.5*, and the threshold for output nodes was set to *0.6*.

First Experiment

Here we built two different search profiles using the same set of original user queries, one for a fruit producer interested in apples (User 1), and one for a user of Apple Macintosh computers (User 2). The network is trained to learn the best query modification for each profile.

Table 2 and Table 3 show the original and the best modified queries with their performance (given by the ratio of relevant to total retrieved documents) for User 1 and User 2, respectively. The results show that our system is capable of learning appropriate query modifications for specific users without requiring the user to provide feedback on any but the original query results. The learned queries here always outperformed the original queries, indicating that the internally derived score estimate adequately represents the relative quality of the queries.

Second Experiment

The search domain for the second experiment is the World Cup soccer tournament. The goal of this experiment is to evaluate the ability of the system to generalize from previous queries to the user's intent with a new query. The domain was chosen to provide sufficient ambiguity in terms of contexts that share common keywords (such as rugby, cricket, American football, or non World Cup soccer). In this experiment, one query at a time was presented and the original queries, the modifications generated by the network based on past queries, and the best learned queries were evaluated. Table 4 compares the performance of all three sets of results for the four original queries used. The results show that the quality of network-generated modified queries significantly improves as more queries are learned, in this case outperforming the original query after the second training query. This is an expected result since at the beginning the network has no knowledge of the user's interests.

User query	User query ratio	Network query	Network query ratio	Best modified query	Best modified query ratio
World Cup	8/17=0.47	Cup	1/20=0.05	World Cup players	14/20=0.7
World Cup match	8/19=0.42	World Cup players match	6/18=0.33	World Cup match results	13/20=0.65
Football match	1/10=0.1	World Cup match results	13/20=0.65	World Cup football matches	14/19=0.737
Football	2/20=0.1	World Cup football	12/19=0.63	World Cup football info	15/20=0.75

Table 4. Ratios of relevant documents to total documents for the user query, network query and modified query

Comparison With Other Approaches

We evaluate the performance of our system by comparing the quality of the documents retrieved through the modified queries learned by the network with the quality of those retrieved by applying other approaches.
We applied the following techniques in order to compare our system:

- Relevance feedback.
- Personalized categories.
- Document clustering.

In all the cases we compare the ratio of documents classified as relevant to the total number of classified documents.

Relevance Feedback

To compare our technique against a method based strictly on relevance feedback, we submit every user's original query to the search engine. For each query, the user classifies the retrieved documents as relevant and irrelevant. We extract the three most representative words of the relevant documents according to their entropy loss value. We create three different expanded queries and select the best one for comparison. These three queries have the following form:

- Original query + best term
- Original query + best term + second best term
- Original query + best term + second best term + third best term

Figure 4 shows the performance of the modified queries learned by our system and the queries expanded using relevance feedback for the domain of apples as fruits. Figure 5 shows the results for the queries related to the soccer World Cup.

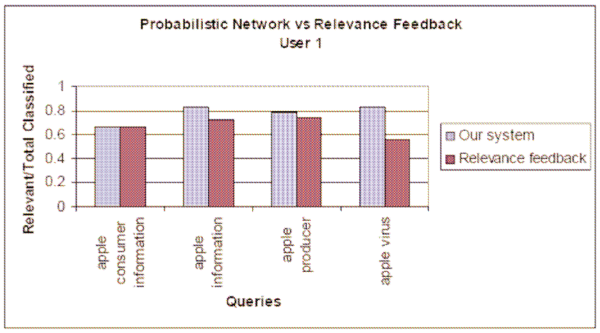


Figure 4. Comparison between our approach and Relevance Feedback using the queries related to the fruit apple.

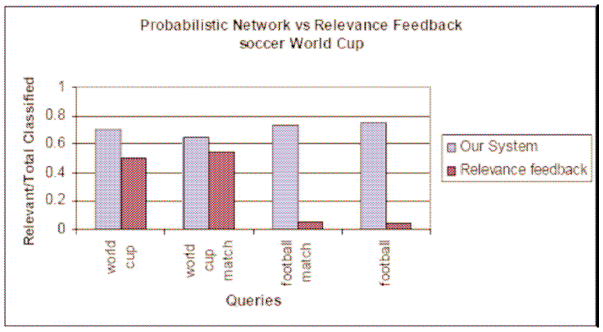


Figure 5. Comparison between our approach and Relevance Feedback using the queries related to the soccer World Cup.

Personalized Categories

To obtain a comparison with a purely category-based, server-side search approach we submit each original query to the search engine and use Google Personalized Beta (http://labs.google.com/personalized) to restrict the retrieved documents to be within a personalized set of categories selected by the user. For the queries associated to the fruit apple, the category Health and the subcategory Nutrition are selected. We select the subcategory Soccer inside the category Sports for the queries referring to the soccer World Cup.
Figure 6 and Figure 7 show the performance for the queries associated to the fruit apple and the soccer World Cup respectively.

5.3 Document Clustering

In this case we use all documents retrieved by a set of queries belonging to the same search domain to create two clusters of documents: relevant and irrelevant. In other words we consider the whole set of documents as if they were retrieved by a single query and classify them as relevant and irrelevant. Similar to the procedure followed for the relevance feedback method, we extract the most

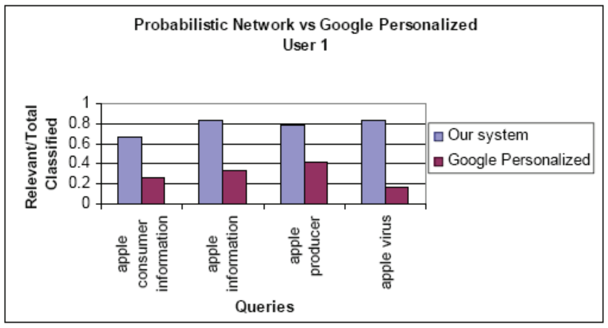


Figure 6. Comparison between our system and Google Personalized Beta using the queries related to the fruit apple.

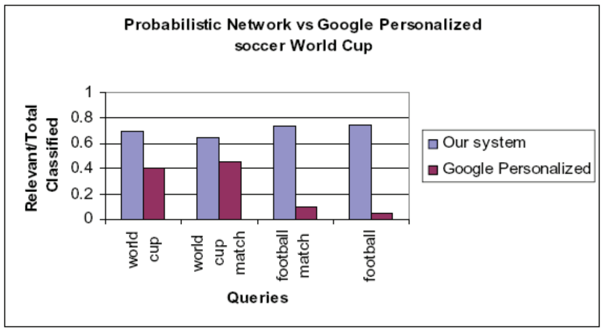


Figure 7. Comparison between our system and Google Personalized using the queries related to the soccer World Cup

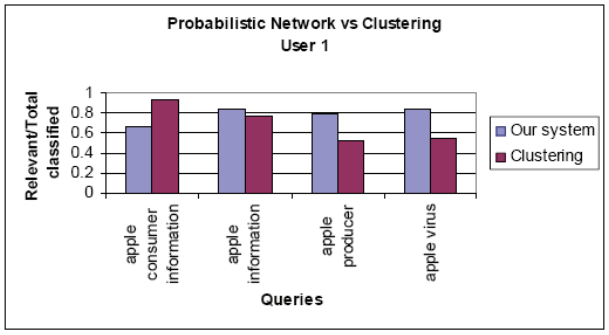


Figure 8. Comparison between our system and Document Clustering using the queries related to the fruit apple.

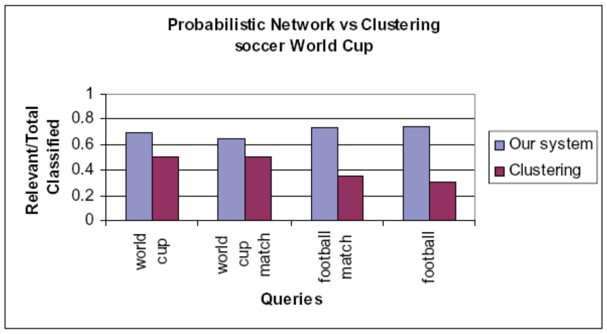


Figure 9. Comparison between our system and Document Clustering using the queries related to the soccer World Cup.

representative words of the relevant documents and use them to create three different expanded queries for each original query and select the best one for comparison. In this way all the queries of the search domain are expanded with the same three words. In this comparison we use the documents retrieved by the queries referring to the fruit apple and use the three most representative words of this search domain to expand every original query. For the soccer World Cup domain the query modifications were learned iteratively meaning that every query modification incorporates the information of the previously learned queries. For the first query (World Cup), we only use the set of classified documents associated to this query; for the second query (World Cup match) we use the documents retrieved by the previous query and the current query; for the third query (football match) we use the documents retrieved by the first two queries and the current query and for the fourth query (football) we use the documents retrieved by all the queries. The comparison of the performance between our system and the document clustering approach is shown in Figure 8 and Figure 9.

According to the comparison results we conclude that the quality of the queries learned by our system and the queries produced through relevance feedback is almost the same for the queries that refer to the fruit apple. However, the performance of the queries produced by our system is better for the domain of the soccer World Cup. In this case the original queries were modified using the information provided by the previously learned queries. In all the cases our system performs better than the personalized version of Google. This may happen because Google only supplies personalized results for the first 10 retrieved documents and because the search categories are defined by Google and they may not exactly match the user's information need. The performance of the queries created with the document clustering approach and the queries formulated by our system are similar to those obtained by applying relevance feedback. Even though in the *"apple fruit"* domain the document clustering approach outperforms our system in some queries, the results for the soccer World Cup domain again suggests that incorporating the user's search profile is useful to increase the performance of the modified queries. It is also important to mention that neither the document clustering nor the relevance feedback techniques provide a mechanism to store the best modification of every original query. This means that every time a query is presented the process of document mining has to be repeated no matter if the query was presented before.

Discussion and Conclusions

We have designed and implemented a system that is capable of learning a personalized search profile from queries created by a particular user and the documents that he/she has classified. The search profile is represented with a probabilistic network that is updated using a gradient-

based learning algorithm. The initial experimental results suggest that the network is able to predict good query modifications as it learns more about the user's search interest. This is especially helpful for ambiguous queries such as football match and football whose original performance is very low compared the queries produced by the network. We also demonstrate that the system is able to learn different search profiles based on the same input queries. This suggests that the quality of search results might be further improved by building separate profiles for different user categories.
One concern that might arise is the complexity of the network. However, this can be reduced by removing unused output and meaning nodes. In addition, when the CPTs become very large, they may be substituted with a neural network which encodes the CPTs in its weights.

References

Babalanovic, M., Shoham, Y.. (1995), "Learning Information Retrieval Agents: Experiments with Automated Web Browsing", AAAI SS IGDR.

Chen, L. and Sycara, K., (1998), "WebMate: A Personal Agent for Browsing and Searching", In Proc. 2nd International Conference on Autonomous agents.

Fellbaum, C.. (1998), "WordNet: An Electronic Lexical Database", MIT Press.

Glover, E.J., Flake, G.W., Lawrence, S., Birmingham, W.P., Kruger, A., Giles, C.L., Pennock, D.M. (2001), "Improving Category Specific Web Search by Learning Query Modifications", SAINT.

Joachims, T., Freitag, D., Mitchell, T. (1997), "Web Watcher: A Tour Guide for the World Wide Web", IJCAI.

Lieberman, H. (1995), "Letizia: An Agent That Assists Web Browsing", IJCAI.

Liu, F., Yu, C., Meng, W. (2002), "Personalized Web Search by Mapping User Queries to Categories", CIKM.

Moukas, A. (1996), "Amalthaea: Information Discovery and Filtering Using A Multiagent Evolving Ecosystem" In Proceedings of the Conference on the Practical Applications of Intelligent Agents and Multiagent Technology.

Pazzani, M., Billsus, D. (1997), "Learning and Revising User Profiles: The Identification of Interesting Websites", Machine Learning 27, 313-331.

Pearl, J. (1988), "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann Publishers, Inc.

Radlinski, F., Dumais, S. (2006), "Improving Personalized Web Search using Result Diversification". In Proceedings of the 29th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 691-692.

Rosenfeld, R. (1996), "Adaptive Statistical Language Modeling: A Maximum Entropy Approach". CMU Thesis.

Salton, G., Buckley, C. (1990), "Improving Retrieval Performance by Relevance Feedback". Journal of the American Society for Information Science, 41(4):288-297.

Shavlik, J., Calcari, S., Eliassi-Rad, T., Solock, J. (1999), "An Instructable, Adaptive Interface for Discovering and Monitoring Information on the World-Wide Web". In Proceedings of the International Conference on Intelligent User Interfaces.

Teevan, J., Dumais, S, Horvitz, E. (2005), "Personalizing Search via Automated Analysis of Interests and Activities". In Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 449-456.

Xu, J., Croft, W.B.. (2000), "Improving the Effectiveness of Information Retrieval with Local Context Analysis" ACM Trans Inf Sys, 18(1):79-112.