Learning Personalized Query Modifications

Erika E. Torres-Verdín, Manfred Huber

Department Of Computer Science and Engineering
University of Texas at Arlington
Box 19015, Arlington TX, 76019
e-mail: etorres.verdin@gmail.com, huber@cse.uta.edu

Abstract

The continuous development of the Internet has resulted in an exponential increase in the amount of available information. A popular way to access this information is by submitting queries to a search engine which retrieves a set of documents. However, search engines do not consider the specific needs of every user and they retrieve the same results for everyone. This suggests the necessity to create a profile that incorporates the search preferences of every user. We present an intelligent system that is capable of learning the search profile of a particular user given a set of queries. We represent the search profile with a probabilistic network that incorporates semantic information and create and implement a gradient-based learning algorithm to update the profile. The ultimate goal of the system is to modify original queries to improve the degree of relevance between the user's search interests and the retrieved documents. The proposed system is a client-side application that is dependent on 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 will retrieve 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 often 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 will retrieve 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. To learn the profile, the system generates a modified query for every user query, submits them to the search engine, and allows the user to classify the results as either relevant or irrelevant. The goal is to obtain information about as many relevant documents as possible. The system then selects the most representative words from the classified documents according to their entropy loss value (Rosenfeld 1996) and includes them into the network. With these words, the system creates a set of new modified queries by randomly combining words in the original query and those obtained from the classified documents. The system submits the modified queries to the search engine and scores the results by comparing the retrieved URLs with the URLs of the classified documents. The network is then trained with the best modified query using a gradient-based algorithm to store the best query modifications and generalize them to new queries.

There have been a number of related research works aimed at helping users find relevant web pages. Query expansion techniques are focused on adding words to an existing query (Xu and Croft 2000). A common characteristic of these techniques is that they do not consider the search history of the user. Significant work has focused on building intelligent agents that assist users while browsing the Internet; some examples are *Syskill and Webert* (Pazzani and Billsus 1997), *WebWatcher* (Joachims, Freitag and Mitchell 1997) and *Letizia* (Lieberman 1995). Other projects have focused on learning a search profile and executing personalized web search such as *Lira* (Babalanovic and Shoham 1995) or *WebMate* (Chen and Sycara 1998). Work in the area of personalized search categories maps the user's queries to a set of categories based on a user profile and general knowledge (Liu, Yu, and Meng 2002). In contrast, the system presented here represents the search profile with a probabilistic network that attempts to infer the information need behind each individual query and suggests personalized modifications.

In the remainder, the paper first describes the network's construction and the procedure to infer modified queries. It then introduces the training algorithm before illustrating the system by learning profiles for different information needs.

Building the search profile

In the approach presented here the search profile of a particular user is represented as a probabilistic network. Every time the user presents a new query, 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 relevant and their respective meanings are added. To select the most representative features, every original query and its associated relevant and irrelevant documents are stored in a database and analyzed to remove HTML tags and stop words. The most significant words are then selected by first eliminating words for which the ratio of relevant to irrelevant documents is worse that the original ratio. Then words are selected whose entropy loss value is above a 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). 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)**.** 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 utility.

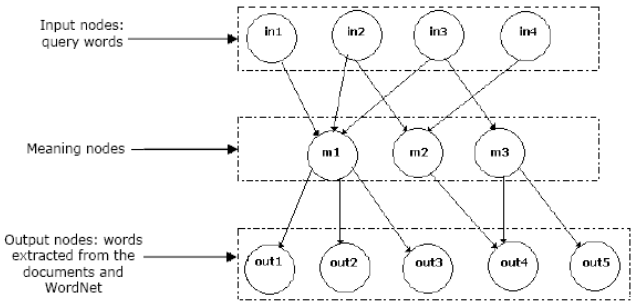


Figure 1. Network topology.

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". According to WordNet "apple" has one and "producer" has three meanings. The ratios of the frequency of each sense 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

Table 1 Frequency ratios of the Meaning Nodes

Word	Sense	Frequency Ratio
Apple	Apple1	2/2
Producer	producer1	9/16
	producer2	5/16
	producer3	2/16

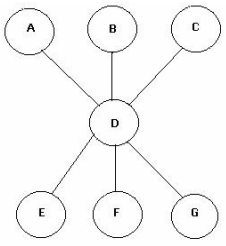
b) Meaning nodes with two or more parents: If two or more input nodes represent words with common meaning, then the value of the word with the highest frequency is used to initialize the meaning node. When the input nodes do not share a common meaning, we only consider the words for which WordNet gives 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". Because WordNet does not provide any senses that link the two input words, the meaning nodes "producer1", "producer2" and "producer3" are initialized using 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 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

The values of the CPTs were initialized using only the information provided by WordNet. The user's queries present new evidence that is used to update the CPTs. Once the network has learned the target output for one or more queries, it 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 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 updates all their CPT entries. Suppose that the network in Figure 2 has learned the best modification for the original queries Q_1=AB and Q_2=BC and the modification for query Q_3=AC has to be predicted.



A	B	C	P(D|ABC) T	P(D|ABC) 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. P*robabilistic network and the CPT for node D.

In this situation, the CPT entry of 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 queries are equally likely and all input nodes are independent given the meaning node D, we get:

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

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

Using $P(D=T|Q_1)$, $P(D=T|Q_2)$, and the assumption that ratios that have never been observed are 1, the unknown ratios of Equations 3 and 4 can be found. $x=P(D=T|Q_3)$ 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 (variances) 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 $\frac{P(B=F\mid D=T)}{P(B=T\mid D=T)}$ and $\frac{P(C=T\mid D=T)}{P(C=F\mid D=T)}$ (Equations 3 and 4) 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 combinations of input nodes. If it is also assumed that the prediction errors are small compared to their expected values, i.e. $\sigma_{J'} << J$ and $\sigma_{K'} << K$, then the variances 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 variance 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 should improve, leading to better query modifications over time.

Learning the search profile

The system learns a search profile from queries of a particular user and the classification that he/she assigns to some of the retrieved documents. For every original query the system internally creates a set of query modifications using the different meanings of the query words and the terms extracted from the classified documents. Every modified query is submitted to the search engine and the retrieved URL's are compared with those of the classified documents to calculate the expected number 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 converts desired output queries to utilities for the output nodes. The main steps 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 and 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: $S=\{s_0,s_1,\ldots,s_N\}$. 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-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 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 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:

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

The best modified query is selected based on its *E(Z)* value.

Learning the best modification

The creation of network training examples from an original query and its best modification follows the following steps:
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 × 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 in the target query. The difference between O_i 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
Current Output: Value (C) ≥ Threshold
Value (C) = Value (C) + α ((Threshold – ε) – Value (C))

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:

$$value(O_i)=value(O_i)+\alpha(Sum(Delta)/size\_of\_Delta)$$

Every time the value of an output node changes, the CPTs are updated using a gradient-based algorithm.

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 output nodes' values is: *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:

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

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:

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

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

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

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 $P(O_i=T|\Pi_{ij}) \in [0, 1]$.
Following this, 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 average the *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 the 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 we define $P(O_i=T)$:

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

where *R* is the number of parents, Π_k, of meaning node M_k and 2^R 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 $P(Mk=T|\Pi kj)$ is:

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

Finally we update $P(Mk=T|\Pi kj)$:

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

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 that 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 for a novel input query. The network creates this output query based on the previously learned queries. In both cases, queries were submitted to a search engine and evaluated according to the ratio of relevant documents to the total number of retrieved documents. In both experiments, the threshold for the extraction of words 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 modifications for each profile.
Tables 2 and 3 show the original and the best-modified queries with their performance for User 1 and User 2, respectively. Here the "modified query score" is the expected ratio of relevant to retrieved documents for a modified query as determined by the metric presented previously. The "modified query ratio" is the real ratio of relevant to retrieved documents for the modified query.

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 information	Apple consumer information produce	7.66/15	5/15	13/15
Apple information	Apple information apples	9.77/18	8/18	15/18
Apple producer	Apple producer growers	9.16/19	8/19	19/19
Apple virus	Apple virus delicious	10.06/18	7/18	15/17

Table 3 Results for the search profile of User 2

User query	Modified query	Modified query score Rel/Total	Original query ratio Rel/Total	Modified query ratio Rel/Total
Apple consumer information	Apple consumer information Mac	9.59/17	8/17	17/17
Apple information	Apple information computer	10.88/18	10/18	18/18
Apple producer	Apple producer Mac	9.40/15	8/15	14/15
Apple virus	Apple virus OS	11.64/17	10/17	17/17

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

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

The "original query ratio" is the ratio of relevant to retrieved documents for the original query. The results show that the 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 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. Here, "user query ratio", "network query ratio" and "best modified query ratio" are the ratio of relevant to retrieved documents for the original query, the query created by the network, and the best modified query, respectively. The results show that the quality of network-generated modified queries significantly improves as more queries are learned, here outperforming the original query after the second training step. This result is expected since in the beginning the network has no knowledge of the user's interests.

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 by a probabilistic network that is updated using a gradient-based learning algorithm. The 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 to 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, *Int. Con. Autonomous Agents*.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*, MIT Press.

Joachims, T., Freitag, D., and 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., and Meng, W. 2002. Personalized Web Search by Mapping User Queries to Categories, *CIKM*.

Pazzani, M. and 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.

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

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