Probabilistic Query Expansion Method Using Recommended Past User Queries

Btihal El Ghali
GSCM-LRIT, Faculty of Science - Mohammed V University, Agdal
Rabat Morocco
btihal.elghali@gmail.com

Abderrahim El Qadi
TIM, High School of Technology
Moulay Ismaïl University
Meknes, Morocco
elqadi_a@yahoo.com

Mohamed Ouadou
GSCM-LRIT, Faculty of Science - Mohammed V University, Agdal
Rabat, Morocco
ouadou55@gmail.com

Driss Aboutajdine
GSCM-LRIT, Faculty of Science - Mohammed V University, Agdal
Rabat Morocco
aboutaj@fsr.ac.ma

***Abstract*—A plenty of Query Expansion techniques have been proposed to solve the problems of information retrieval systems, but new challenges has been introduced for these methods of expansion of user queries because of the rapid growth of the size of the Web collection. In this paper we have focused our attention on the improvement of the precision of the user query by a Probabilistic Query Expansion Method. Our approach consists on the use of information contained in query logs, which includes past user queries and their clicked documents, for providing high-level recommendations. The experiment results shows that the precision of the short queries increases even if we add a few terms, but the same number of terms added don't affect the long queries precision as much as for the short queries.** *(Abstract)*

***Index Terms*— Query Expansion; Query recommendation; Query logs; Probabilistic Method.**

I. INTRODUCTION

With the enlargement of the accessibility of the Internet, the amount of data available online at low cost, and the number of users of web sites has increased dramatically. In recent years, the overgrowth of information and the lack of organization in their resources on the World Wide Web have made it difficult for web users to access the information they need. Above all, the queries that contain ambiguous terms can retrieve documents that do not correspond to the field of research of the user and can lead him to error if he does not have specific information about what he is searching or if he is not expert in the domain of his research.

In addition to these issues, the vocabulary varies considerably from an author to another, without neglecting the fact that users generally tend to not use the same terms contained in documents as search terms.

It is also generally observed that web users submit very short queries to search engines and that the average length of web queries is less than two words [1]. The short queries usually do not have enough words to cover useful search terms and thus negatively affect the performance of the research.

To overcome this problem and improve the effectiveness of information retrieval, the researchers used the Query Expansion methods that involve the reformulation of queries by adding new terms to the initial query of the user, and they also proposed Query recommendation methods that suggest the nearest queries to the initial query.

Traditional approaches of query expansion were focused on expanding the query with terms extracted from various sources of information such as a thesaurus like WordNet, or from the top-ranked documents used in relevance feedback techniques.

However, these methods of expansion were limited in the extraction of expansion terms from a document collection, and have not used information about interactions and user choices; this is the case of the expansion based on the use of query logs.

II. DESCRIPTION OF THE QUERY EXPANSION METHOD

The query logs stored in web servers have been widely studied with various techniques of web mining in recent years, for improving the effectiveness of search engines and their usability [2] [3]. Such studies mined the logs to improve numerous search engines capabilities, such as query suggestion, query classification, ranking, targeted advertising, etc. Given a log containing a large number of queries, it is straightforward to build a surrogate for each document in the collection, consisting of the queries that were a close match to that document [4].

In what follows, we consider that a document clicked-on for a query is a relevant document to this query.

In our approach, we assume that the relationship between queries and their relevant documents is a bipartite graph (Fig. 1). As a collection of test we used a database from the standard collection Smart: database CISI. If two queries are linked with the same relevant document, we believe that these two queries are associated with each other in some way, and the terms in the associated queries can be used as candidate terms for query expansion, we may also use the terms of the relevant documents of an associated query in the process of expanding its associated query.

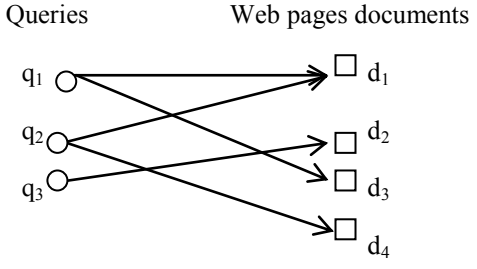

Fig. 1. Query – URLs representation as a bipartite graph

For the exploitation of the graph information, we represent each query as a document vector $q_i=\{d_1, d_2, \dots d_N\}$, whose elements describe the presence or absence of a document in all relevant documents to this;

For a query qi if the document dj is relevant to the query qi then $d_j=1$, otherwise $d_j = 0$.

We also represent each query as a vector of the terms contained in all the queries $q_k=\{t_1, t_2, \dots t_M\}$.

For a query q_k if the term t_l is present in the query then $t_l=1$, otherwise $t_l=0$.

These queries are used as input for the query recommendation algorithm. Before describing this algorithm, we will define the expressions of similarity and the weighting functions that we will use in our Query Recommendation Algorithm.

In previous works [9], we made a comparison of the best known similarity measures, and a comparison of the most commonly used weighting functions. Thus, in this work we will use those judged the best in our query recommendation process.

A. *Weighting function : LTC*

The weighting is a fundamental step in most approaches proposed in information retrieval [5].

The weighting functions are usually used to measure the weight of a term in a document. But in our approach, we calculate the weight of a document or a term for a query, which reflects the degree of relevance of this document or term for this query.

In this work, we chose the weighting scheme LTC that is constructed by combining two factors:

- A local weighting factor (number of occurrences of the term in the document), which reflects the importance of the term within a document.
- A second factor, of global weighting, which is the value of the importance of the term throughout the collection whose formula is: $\log(N/n_i)$ where N is the size of the collection and n_i the number of documents where appears the term t_i.

The formula of this weighting function for a document $D_j = [r_{1j}, r_{2j}, \dots, r_{mj}]$ and a term t_i is as follows [6]:

$$r_{ij} = \frac{\log(tf_{ij}+1)\times idf_{ij}}{\sqrt{\sum_{k=1}^{m}[\log(tf_{kj}+1)\times idf_{kj}]}} \qquad (1)$$

By using the logarithm of the local weighting factor, which represents the frequency of appearance of a term in the document, this expression reduces the effects of large differences in frequencies.

B. *Similarity measure : Cosine Similarity*

The purpose of the computation of similarities between the queries, represented as a document vectors, is the search for common documents, and then make the link between the queries and their relevant documents. For those represented as a term vectors, it is the search for common terms, to make the link between the queries and the terms that are present in these queries.

We chose to use the cosine similarity which is a measure that calculates the similarity between two vectors by determining the angle between them, because it was judged as the best similarity measure in a previous work that we have done.

Two objects (documents) are similar if their vectors are confounded [7].Otherwise, the two objects are not similar and their vectors form an angle ($\vec{q}_i, \vec{q}_j$), whose cosine is the value of the similarity.

The expression of similarity based on cosine is:

$$Simc(\vec{q}_i, \vec{q}_j) = \cos(\vec{q}_i, \vec{q}_j) = \frac{\vec{q}_i \times \vec{q}_j}{\|\vec{q}_i\| \times \|\vec{q}_j\|} \qquad (2)$$

C. *Query Recommendation Algorithm*

In our research, we applied an algorithm that finds the appropriate group of related queries with the input query and classify them according to their relevance to it. The interest of a related query is measured by its rank (score), which is obtained by combining the following concepts:

a. The similarity of the query, that is measured using the expressions of similarity defined above.

b. The support of the query, this is a measure of the relevance of the query in the collection.

The similarity and the support of a query can be normalized and then combined linearly to generate the rank of the query.

The Query Recommendation Algorithm works as follows:

-1- We represent each query in the space of queries by a query vector $Q_j=\{D_1^{(j)}, D_2^{(j)}, \dots, D_n^{(j)}\}$ and $Q_j=\{T_1^{(j)}, T_2^{(j)}, \dots, T_m^{(j)}\}$, where $D_i^{(j)}$ represents the weight of the i^{th} document in the query vector, and $T_i^{(j)}$ represents the weight of the i^{th} term in the same query vector. These weights are defined by classic weighting measure Ltc:

$$D_i^{(j)} = \frac{\log(tf_i^{(j)}+1)\times idf_i^{(j)}}{\sqrt{\sum_{k=1}^{m}[\log(tf_k^{(j)}+1)\times idf_k^{(j)}]}} \qquad (3)$$

with: $idf_i^{(j)}=\log(N/n_i)$

Where $tf_i^{(j)}$ represents the frequency of the i^{th} document in the query vector Q_j, N is the total number of queries in the collection and n_i is the number of queries for which the i^{th} document is relevant.

We calculate each $T_i^{(j)}$ with the same expression.

-2- We calculate the similarities between our initial query and all the past queries contained in the log by using the similarity measure: Cosine.

-3- We calculate the support of each query q_j, which we measured as the ratio of the number of relevant documents to the query q_j and the total number of relevant documents for all the queries of the collection, as follows:

$$Sup(q_j) = \frac{N_{RDs}^{(q_j)}}{\sum_{k=1}^{N_q} N_{RDs}^{(q_k)}} \quad (4)$$

Where: N_q is the number of queries and $N_{RDs}^{(qj)}$ is the number of relevant documents to the query q_i.

Then we compute the support for these queries represented as a term vectors.

-4- We classify the queries based on their similarities and their supports using the two representations of each query.

The ranking score base on document vectors is as follow:

$$Rank(q_j) = [\alpha \times Sim_D(q_j, q_i) + (1-\alpha) \times Sup_D(q_j)] \quad (5)$$

Our contribution in this work takes into account also the use of the queries terms in the recommendation algorithm, and the ranking score based on term vectors is measured as follow:

$$Rank(q_j) = [\beta \times Sim_T(q_j, q_i) + (1-\beta) \times Sup_T(q_j)] \quad (6)$$

In this Query Recommendation Algorithm, we will use the combination of these two ranking scores. Thus, the recommendation score of the query q_j for the query q_i is measured by:

$$Rank(q_j) = [\alpha \times Sim_D(q_j, q_i) + (1-\alpha) \times Sup_D(q_j)] + [\beta \times Sim_T(q_j, q_i) + (1-\beta) \times Sup_T(q_j)] \quad (7)$$

With β and $\alpha \in [0,1]$, they are used for normalization.

D. Query Expansion Method

Our query expansion method takes as inputs the terms of the user query (initial query), and the candidate terms for the expansion of this query: the terms extracted from relevant documents (RDs) of the initial query, the best recommended queries and their relevant documents (RDs). These terms are classified according to the whole query, using a probabilistic method.

The expansion method that we have chosen to use in combination with the query recommendation algorithm is the method proposed in [8] and it includes three steps:

-1- We calculate the weight of the terms of the documents $w_i^{(d)}$ using the normalized formula of TF-IDF :

$$w_i^{(d)} = \frac{tf_i^{(d)} \times idf_i^{(d)}}{\sum_{\forall D \in S} tf_i^{(d)} \times idf_i^{(d)}} \quad (8)$$

Where S is the set of documents used for expansion (recommended queries are also considered as documents), $tf_i^{(d)}$ the frequency of the i^{th} term in the document D, and $idf_i^{(d)} = \log(N/n_i)$ with N as the number of documents in the set S and n_i as the number of document containing the i^{th} term.

The weights of query terms $w_i^{(q)}$ are calculated similarly.

-2- We compute the probabilistic correlations between every query term and documents (the RDs, the most related queries and their RDs) terms. The expression of the probabilistic correlation used is as follow:

$$P(w_j^{(d)} | w_i^{(q)}) = \sum_{\forall D \in S} P(w_j^{(d)} | D) \times P(D | w_i^{(q)}) \quad (9)$$

$P(w_j^{(d)} | D)$ is the conditional probability of occurrence of the j^{th} document term if the document D is selected.

$P(D | w_i^{(q)})$ is the conditional probability of the document D being clicked when the i^{th} query term appears in the user query. These two conditional probabilities can be determined as follow:

$$P(w_j^{(d)} | D) = \frac{w_j^{(d)}}{\max_{\forall t_k \in D} w_k^{(d)}} \quad (10)$$

$$P(D | w_i^{(q)}) = \frac{f_i^{(q)}(w_i^{(q)}, D)}{f^{(q)}(w_i^{(q)})} \quad (11)$$

Where

$f_i^{(q)}(w_i^{(q)}, D)$ is the number of the query sessions in which the term $t_i^{(q)}$ appear in and for which the document D is clicked-on.

$f^{(q)}(w_i^{(q)})$ is the number of the query sessions (the queries) that contain the term $t_i^{(q)}$.

-3- By combining the probabilities of all query terms, we can calculate the cohesion weight of a document term, which present the relationship (correlation) of the document term with the whole query.

The cohesion weight of a document term $t_j^{(d)}$ for a user query Q is measured by the expression:

$$CoWeight_Q(w_j^{(d)}) = \ln\left(\prod_{t_i^{(q)} \in Q} \left(P(w_j^{(d)} | w_i^{(q)}) + 1\right)\right) \quad (12)$$

This method returns a list of weighted terms. The top-ranked terms can be selected as expansion terms for the initial user query.

III. EXPERIMENTAL RESULTS

A. Query recommandation for short and long queries

In our experiments, we realized a java application to do the calculations. As a collection of test we used a database from the standard collection Smart: database CISI (Collection of document abstracts in library science and related areas). This collection is provided with queries and their ground of truth (a list of documents relevant to each query). Table 1 lists statistics about the database used.

TABLE I. STATISTICS ABOUT THE COLLECTION SMART

# document	# queries
1460	111

As we mentioned in Section 2, we used the information contained in this data set, first to research the recommended queries to each input query. The input queries that we used are four short queries (contain less than 5 terms) and four long

queries (contain more than 5 terms) that have the highest numbers of relevant documents in the database CISI (table 2).

TABLE II. INPUT QUERIES

N°Query	20	27	30	34	15	24	32	44
Terms number	4	5	4	5	10	12	14	26
Number of RDs	144	115	134	38	82	52	117	155

For our query expansion method, in the first step, we built a document-weighted vector and a term-weighted vector for each query using the function LTC, and we calculated the scores of all queries with each input query using the Query Recommendation Algorithm.

To show the difference between the influence of each score (the score based on term vectors and the one based on document vectors) on the global score (the score based on document and term vectors), we built a line graphs (Figure 2 and Figure 3) using the scores of the best recommended query for every input query (The scores values are displayed on the table 3 and table 4).

Table 3, shows the three different scores for the best recommended queries of the four short input queries. We chose to consider the global score for selecting the best recommended query.

TABLE III. THE DIFFERENT SCORES FOR THE BEST RECOMMANDED QUERY OF SHORT QUERIES

N° Query	Best Recommended Queries	Score based on document vectors	Score based on term vectors	Score based on document and term vectors
20	**104**	0,015	0,532	0,547
27	**28**	0,074	0,701	0,775
30	**25**	0,013	0,203	0,216
34	**27**	0,153	0,587	0,769

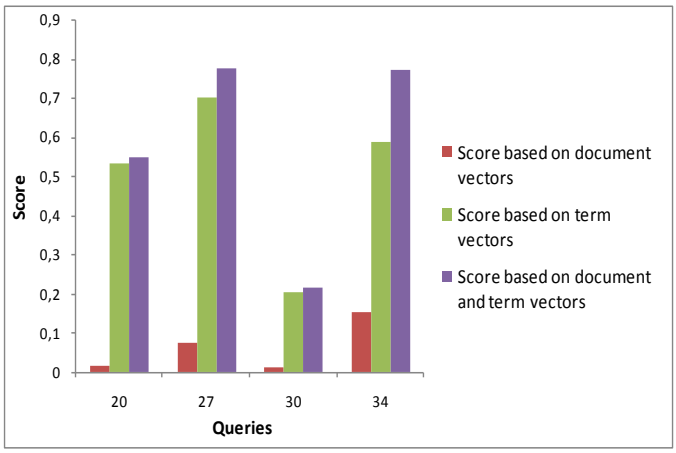

Fig. 2. The comparaison of the scores of the best recommanded query for each short query

The figure 2 shows that the score based on terms influence the global score more that the score based on documents, their values are very close, and that the score based on documents have very minimal values.

In contrast, table 4 shows the three different scores for the best recommended queries of the four long input queries.

TABLE IV. THE DIFFERENT SCORES FOR THE BEST RECOMMANDED QUERY OF LONG QUERIES

N° Query	Best Recommended Queries	Score based on document vectors	Score based on term vectors	Score based on document and term vectors
15	**25**	0,008	0,644	0,652
24	**21**	0,136	0,800	0,936
32	**46**	0,341	0,033	0,375
44	**45**	0,051	0,220	0,271

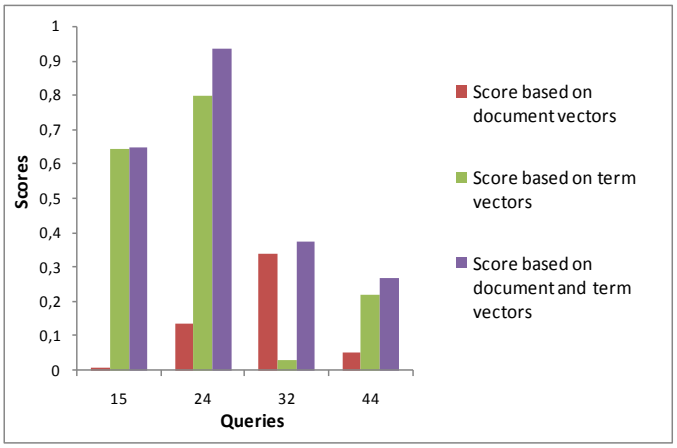

Fig. 3. The comparaison of the scores of the best recommanded query for each long query

In figure 3, we notice that the score based on terms influence the global score most of the time except for the query number 32 whose score based on document vectors is more dominant in the global score, that can be explained by the fact that the queries 32 and 46 have a lot of common relevant documents and that influenced the value of association between them even if they don't have many common terms.

We note that the score based on term vectors is the most dominant for most of the queries whether short or long.

B. Query Expansion

In the second step, we took the four best recommended queries for each input query to use them in the Probabilistic Query Expansion method. The table 5 shows the input queries and their best four queries.

TABLE V. THE INPUT QUERIESAND THEIR BEST RECOMMANDED QUERIES

N° Query	Best 4 Recommended Queries
Short queries	
20	104, 84, 65, 37
27	28, 34, 6, 100
30	25, 13, 15, 44
34	27, 19, 4, 28
Long queries	
15	25, 13, 35, 26
24	21, 66, 31, 7
32	46, 8, 97, 1
44	45, 42, 92, 7

Before using the probabilistic method to expand the input queries, we had to do a pretreatment of the terms contained on the queries and the relevant documents used in the process of expansion. This pretreatment is divided on two steps, the first one is the elimination of the stop words and the second one is the stemming of the terms. Then we expanded each of the eight input queries by adding the five terms that have the highest Coweight relatively to these queries.

To evaluate the performance of the retrieval system, we used the Interpolated Average Precision (IAP): we did a series of experiments to investigate the effects of the number of recommended queries used to expend the input queries on the interpolated average precision. We examined the performance by using only the first best recommended query (RQ), its RDs and the RDs of the input queries, and then we added the second and its RDs, the third and its RDs and finally the fourth best recommended query and its Rds. Figure 4 and figure 5 show the impact of the number of the recommended queries used on the IAP.

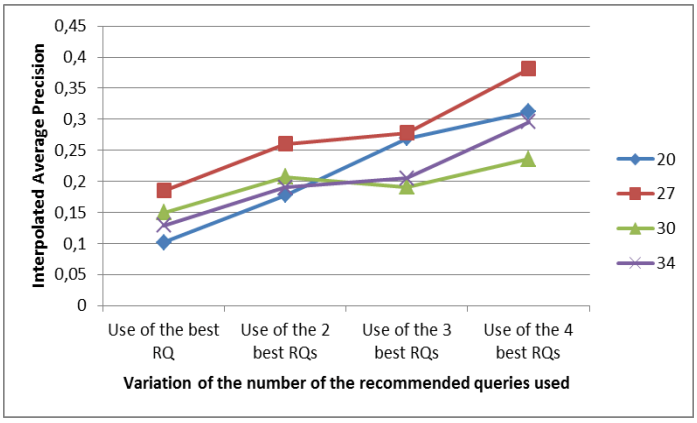

Fig. 4. The impact on short queries of different number of recommended query used for expansion

The experiments illustrated in figure 4 show that the IAP is increasing while we add more recommended queries to the process of expansion, because the precision of the input queries increases more and more.

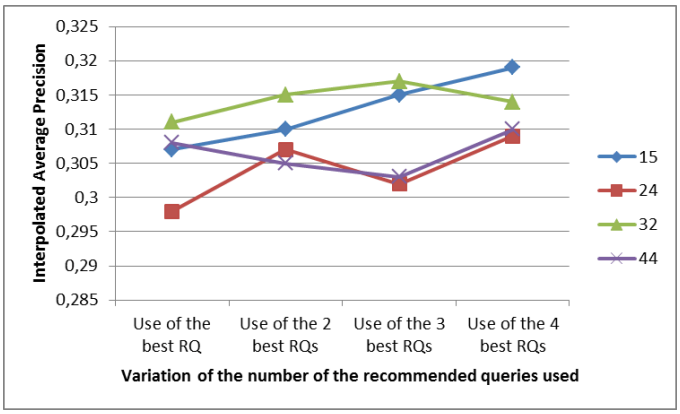

Fig. 5. The impact on long queries of different number of recommended query used for expansion

The figure 5 shows ups and downs in the IAP of the long input queries, but when we refer to the table 6, we notice that that the value of the IAP is almost unchanged from an extended query to another, that amounts to the number of term contained in the input queries.

We can assume that for the long queries the expansion by two terms is not enough that's why the performance wasn't improved.

IV. CONCLUSION AND FUTURE WORK

In this paper we have presented a method for query recommendation and query expansion, based on the relevant documents and the terms contained in the user past queries. We did the experimentations with the four short queries and four long queries that have the largest number of relevant documents in the database CISI of the Collection Smart. The results show that most of the time the score of a recommended query based on terms is more important than the score based on documents, and that by adding two terms of expansion for every input query, the short queries precision has increased while using more recommended queries, but concerning the long queries the expansion by the same number of terms didn't had a very important impact on the interpolated average precision.

As future work, we propose to improve the approach developed in this work, and apply it in data set extracted from a real search engine query log.

REFERENCES

[1] J. Wen, J. Nie and H. Zhang. "Clustering User Queries of a Search Engine", WWW10, Hong Kong, May 1-5, 2001.

[2] Z. Kunpeng, W. Xiaolong, L. Yuanchao. "A new query expansion method based on query logs mining", International Journal on Asian Language Processing, China, Vol 19 Number 1: 1-12, 2009.

[3] R. Baeza-Yates, C. Hurtado and M. Mendoza, "Query Recommendation Using Query Logs in Search Engines", LNCS 3268, Springer-Verlag Berlin Heidelberg, pp. 588-596, 2004.

[4] H. M.Zahera, G. F. El Hady, W. F. Abd El-Wahed. "Query Recommendation for Improving Search Engine Results", Proceedings of the World Congress on Engineering and

Computer Science 2010, San Francisco, USA, Vol I, WCECS 2010, October 20-22, 2010.

[5] H. TEBRI. “Formalisation et spécification d’un système de filtrage incrémental d’information”, Thèse, Institut De Recherche En Informatique De Toulouse, l’Université Paul Sabatier de Toulouse, 15 décembre 2004.

[6] A. Kjersti and L. Eikvil. “Text Categorisation: A survey”, Norwegian Computing Center, Norway, June 1999.

[7] T. Slimani, B. Ben Yaghlane et K. Mellouli. “Une extension de mesure de similarité entre les concepts d’une ontologie”, SETIT 2007 (4rth International Conference: Sciences of Electronic, Technologies of Information and Telecommunications), TUNISIA, March 25-29, 2007.

[8] H. Cui, J. Wen, J. Nie and W. Ma. “Probabilistic Query Expansion Using Query Logs”, WWW2002, Honolulu, Hawaii, USA, May 7-11, 2002.

[9] A. El Qadi, B. El Ghali, D. Aboutajddine : Query Expansion Based Past User Queries. Congrée EGC'2011, pp. 249-258, 23-25 Novembre Ensa, Tanger, Maroc.