On The Reuse of Past Searches in Information Retrieval: Study of Two Probabilistic Algorithms

Claudio Gutiérrez-Soto, Gilles Hubert

On The Reuse of Past Searches in Information Retrieval:
Study of Two Probabilistic Algorithms

Claudio Gutiérrez-Soto, IRIT, Université de Toulouse, Toulouse, France & Departamento de Sistemas de Información, Universidad del Bío Bío, Concepción, Chile

Gilles Hubert, IRIT, Université de Toulouse, Toulouse, France

ABSTRACT

When using information retrieval systems, information related to searches is typically stored in files, which are well known as log files. By contrast, past search results of previously submitted queries are ignored most of the time. Nevertheless, past search results can be profitable for new searches. Some approaches in Information Retrieval exploit the previous searches in a customizable way for a single user. On the contrary, approaches that deal with past searches collectively are less common. This paper deals with such an approach, by using past results of similar past queries submitted by other users, to build the answers for new submitted queries. It proposes two Monte Carlo algorithms to build the result for a new query by selecting relevant documents associated to the most similar past query. Experiments were carried out to evaluate the effectiveness of the proposed algorithms using several dataset variants. These algorithms were also compared with the baseline approach based on the cosine measure, from which they reuse past results. Simulated datasets were designed for the experiments, following the Cranfield paradigm, well established in the Information Retrieval domain. The empirical results show the interest of our approach.

Keywords: Collection Simulation, IR, Probabilistic Algorithm, Reusing Past Queries

INTRODUCTION

A wide range of research lines provide support to Information Retrieval (IR), such as indexing techniques, weighting schemes, matching functions, formal models, and relevance feedback. Over a considerable period of time the retrieval process was applied in a standardized manner for all users, who used a given IR system (IRS). Therefore, given a common query for different users, the final result of this query was the same for each user. Then, new subfields have appeared such as personalized IR, contextual IR, and collaborative IR. These subfields aim at exploring new approaches that adapt the returned results according additional aspects such as the profile of the user who submitted the query, the context (e.g., location) in which the query was submitted, and other users having the same information need.

Nowadays, most of IRSs store data that are associated with the queries. Nonetheless, few approaches take advantages from past search results. Our main assumption is that a set of

past search results can be useful to answer new queries. Some approaches exploit the previous searches in a customizable way for a single user. Nevertheless, the use of previous searches is based mainly on either repeated queries or query reformulations inside search sessions. Approaches that address past searches collectively are less common. This type of approaches is catalogued in collaborative information retrieval. An example of such research is provided by (Hust, 2004), which proposes query expansion based on past searches for new submitted queries. The assumption is that a user may benefit from search experiences of other users who share the same information need, to reformulate her/his query.

The approach proposed in this paper leverages past searches by using past results to build the results returned for new submitted queries. The proposed approach does not aim at improving queries previously submitted by a user (i.e., repeated queries). On the contrary, the approach concerns new queries submitted by a user (i.e., not yet submitted by the user in the past) that were submitted by other users in the past. It aims at introducing a collaborative aspect by taking advantage of previous searches of other users. The approach is based on two separated aspects: a storage aspect and a retrieval aspect. This paper focuses on the retrieval aspect, where relevant documents are selected from the result list of the most similar query to build the response for a new query. It presents, among others, a study of two possible probabilistic algorithms for the document selection.

The IR literature holds a lot of probabilistic approaches, some of them proposed a long time ago. The probabilistic algorithms proposed in this paper correspond to the Monte Carlo category (Burgin, 1999). These algorithms assign the highest likelihood to top-ranked documents in past result lists. The assigned likelihood decreases according to descending ranks of documents in past result lists. The major advantages of these algorithms are the next: they are quite easy to implement and they do not require learning step.

Validation of our approach implies experiments with multiple objectives. A first objective is to compare our approach based on the reuse of past queries with the baseline approach of information retrieval from which it reuses past results. A second objective is to study the two proposed algorithms, which can be used in our approach for the construction of new result lists from past results, in different environments.

Traditionally, a significant portion of research in IR has been related to system evaluation. Evaluation is devoted essentially to effectiveness (i.e., result accuracy), which is obtained by measuring precision (i.e., fraction of relevant documents in the retrieved documents). Although a large amount of IR collections can be found currently, it is a difficult task to find ad-hoc collections to evaluate approaches based on past queries. Most existing IR collections, such as the well-known TREC collections, are composed of very dissimilar queries. With such collections, evaluating an approach looking for similar past queries has no sense. Some collections using query logs of search engines contain similar queries, but these similar queries are mostly repeated queries or query reformulations submitted by the same user. These collections are designed to particular IR tasks such as session detection (Kanoulas et al., 2010; Gayo-Avello, 2009) or result personalization (http://www.kaggle.com/c/yandex-personalized-web-search-challenge), which have different objectives compared to the one tackled by our approach. Such collections cannot be used directly for our objective. Moreover, collections based on query logs do not provide the documents in results lists that are considered relevant by the user who submitted the query. Such information is required to compute precision for system evaluation. User clicks on returned documents are often provided in this type of collections. However, such information cannot be used directly as judgments of absolute relevance (Cen et al., 2009). The usual construction of an appropriate IR collection is very costly in time and efforts, though solutions to reduce the cost have been proposed (Sanderson, 2010; Alonso & Mizzaro,

2012). A solution chosen by some approaches was to adapt existing IR collections to evaluate their approaches, for instance for approaches based on past queries that responded to other issues (Cetintas et al., 2011).

A raising alternative for evaluation of IR approaches is simulation (Azzopardi et al., 2011). Following this line, we propose a method to simulate an IR collection to evaluate the performance of our approach based on the use of similar past queries. Simulation includes document collections, queries, as well as users' judgments, following the Cranfield paradigm, well established in the IR domain. We apply this method to build different simulated datasets.

This paper is organized as follows. In Section 'Related work', works on past searches and simulation in the IR context are presented. Then, Section 'Principles for collective reuse of past searches' presents the overall retrieval process capitalizing on past queries. Section 'Approach for reusing past queries' details two probabilistic algorithms that can be used in our approach for document selection in past searches. Mathematical definitions are given. Then, in Section 'Simulation of information retrieval collections', we present our approach to simulate an IR collection, in the context of past search results. In Section 'Empirical results', experiments and results validating our approach are described. We compare our approach with the traditional retrieval approach it reuses past results and we compare the two probabilistic algorithms. Finally, conclusions and future work are presented.

RELATED WORK

Past Searches in IR

Two major categories of approaches that deal with the use of similar past queries can be found in the modern IR literature. The first category is based on user sessions when users are logged in to some systems. Considering a user session, collection of various data is possible at an individualized level, such as user identification, historical records of queries and retrieval documents, date, session duration (i.e., time) among others. Such data are then used for personalization. The second category is based on anonymous data, and thus can be applied in systems where users are not logged in.

The area of Personalized Information Retrieval (PIR) gathers several approaches that are founded on past search results (Ghorab et al., 2013). (Steichen et al., 2012) presents a survey of the main techniques and their impacts in PIR and Adaptive Hypermedia (AH) technologies. These techniques are examined in several activities of the retrieval process: query adaptation, adaptive retrieval as well as adaptive result composition and presentation. The main conclusions related to past queries, concern query adaptation and adaptive retrieval. For query adaptation, systems either require user involvement such as ad-hoc relevance judgments or create user models that involve keyword and category classifications. Finally, in adaptive retrieval, users' past searches are used to disambiguate a query, by matching terms with several categories.

(Fitzpatrick & Dent, 1997) proposes an approach for automatic query expansion relying on past queries. The fundamental conjecture is that the top documents recovered from result lists of similar past queries are a good source to improve automatic query expansion. Experimental scenarios on TREC collections show better top-precision for the past-queries feedback compared with standard top-document feedback and with no feedback.

(Cui et al., 2003) proposes a method for automatic query expansion based on logs. User logs are exploited to extract implicit relevance judgments, considering a document that was chosen by a user as relevant. Correlations between query terms and document terms are then extracted and used for expansion of new queries. A set of experiments, which take into consideration both long and short queries, show that the log-based query expansion provides important performance improvements. Final results show that query expansion leads to better effectiveness for short queries than for long queries.

(Shen & Zhai, 2003) proposes to combine the results from past queries with the result of a new query in an active session of a user (i.e., the set of queries corresponds to the same user, in the same context). The ranks of the documents in the different results are averaged to order the final result. In (Shen et al., 2005), implicit feedback information is used to improve retrieval accuracy at an individual level. This feedback relies on previous queries and clickthrough history within the same session. Four statistical language models are studied (e.g., Bayesian interpolation), which use the feedback information to rerank the result returned for a new submitted query applying a traditional IR model. Experiments on TREC data show that clickthrough history is implicit feedback that particularly leads to substantial improvement of retrieval performance.

Another idea, proposed in (Song & Mayeng, 2012), is to act on term weighting schemes, which are a key aspect in IR. It proposes a novel term weighting method based on past retrieval results. The main assumption is that the general importance of a term depends on its role in the past retrieval sessions. The method considers that the queries, their corresponding results (i.e., documents), and their relevance judgments are available. A term weight depends on the frequency values for the relevant and non-relevant document distributions in the past, as well as the rankings and similarity values of the relevant and non-relevant documents. Experiments on TREC collections show that the proposed weighting scheme leads to improvements of retrieval effectiveness. Furthermore, these improvements are obtained with only a small number of terms from a small number of past queries.

In the area of federated text search (i.e., a unified search interface for multiple search engines of distributed text information sources), (Cetintas et al., 2011) uses the past queries for resource selection. A new resource selection technique is defined, which estimates the utilities of available information sources for a specific user query based on a weighted combination of results of similar past queries.

More general studies on query logs intend to analyze the usage of search engines, especially regarding query formulation. For instance, (Jansen et al., 2000), analyzing Excite97 query log, found that when users redefine their queries, 34.7% of modifications corresponded to word substitutions; meantime 19% of modifications were single word additions and 9.5% double word additions. The results showed that users prefer to redefine queries by adding terms. (Teevan et al., 2007) explored issues of re-finding using Yahoo query log. The analyses revealed that repeat searches and repeat clicks were very common. They revealed also that 7% of repeat queries were issued by different users.

In contrast to PIR approaches, approaches of collaborative information retrieval (CIR) use past searches collectively. However such approaches are less common. Several query expansion methods based on past searches are proposed in (Hust, 2004) to rewrite a new submitted query. A first method rewrites the new query as a sum of selected relevant documents of existing old queries with similarity to the new query. Another expansion method reconstructs the new query as a linear combination of existing old queries. Finally, other methods reweight document terms or query terms based on the relevant documents of the most similar queries.

In contrast with (Hust, 2004), we propose a CIR approach that uses past searches at the result level rather than at the query level. Our approach aims at building the result for a new query submitted by a user from results of similar queries submitted previously by other users.

Simulation in IR Evaluation

The use of simulations is considered as a research line offering a great potential for the field of IR (Azzopardi et al., 2011; Clough & Sanderson, 2013). Simulation is present in several works in the IR literature.

In (Huurnink et al., 2010) simulators are designed and validated to generate queries and relevance judgments for evaluation of purchase-query retrieval systems. A simulator generates purchase-query pairs, consisting of a query and

an associated relevant purchased document. The large variation in simulator assessments allowed creating artificial testbeds for retrieval experiments. However, no simulator produced an evaluation testbed leading to system rankings like with a real-word gold standard.

(Tague & Nelson, 1981) proposed an algorithm to simulate user relevance judgments for evaluation of bibliographic retrieval systems. Experiments for validating simulation with respect to real system were done on two bibliographic retrieval test collections. Nevertheless, the observed differences led to the conclusion that further model development was needed.

(Azzopardi et al., 2007) focuses on methods for building simulated known-item topics. A detailed analysis of existing generation models on six European languages explores factors that may influence the generation of the known-item topic. A model based on improved document and term selections allows generating simulated known-item topics that are comparable to real known-item topics.

Query reformulation techniques based on query logs have been proposed to improve retrieval effectiveness. In cases where query logs are not accessible, (Dang & Croft, 2010) proposes to use anchor text for query log simulation. Experiments on TREC collections show that an anchor log extracted from a web collection can be a competitive substitute to a real query log.

In the context of federated text search, (Cetintas et al., 2011) simulated past queries from existing TREC collections to carry out experiments on resource selection. In a first approach, past queries were generated from the titles of some top ranked documents returned for a test query. In a second approach, past queries were generated from queries by randomly removing some terms. However, no directions were given for obtaining relevance judgments associated to simulated queries.

Eventually, with the exponential growth of the Web, simulation can give interesting approximations of performance on the Web (Baeza-Yates et al., 2005; Marin et al., 2010).

PRINCIPLES FOR COLLECTIVE REUSE OF PAST SEARCHES

Most information retrieval systems do not differentiate users submitting queries, i.e., if no changes occur in the source of information, a given query always leads to the same result list. One reason is that information retrieval systems usually have no information about users submitting queries. Some approaches propose to personalize the results for each user and each query depending on information about the user. Such approaches rely on user preferences explicitly given by users or deduced from the interactions of users during their previous searches (Ghorab et al., 2013). Other approaches use past searches of a user to rerank the results of queries previously submitted by the same user (i.e., repeated queries). Nevertheless, such approaches rely on a partial and individual use of past searches.

However, queries submitted by a given user in the past constitute a valuable source of information, not only for her/him but also for other users having the same information need. Few systems exploit this source of information in a collective or collaborative manner. This type of issues is part of collaborative information retrieval (CIR). For example, Hust (2004) proposes a collaborative approach of query expansion based on past queries (and their associated results) submitted by different users.

The approach we propose in this paper aims to add a collaborative dimension to the retrieval process in the step of result construction. The approach is based on two separated components: a storage component and a retrieval component.

On one hand, the storage component aims at recording the submitted queries and their computed results. A basic approach consists in storing, for each search, the lists of terms defining the query and the document identifiers of its result list with indications about relevant documents for the user who submitted the query. Such principle is quite similar to query logs usually handled by search engines (Potey et al., 2013). In the absence of explicit relevance judgments, user clicks could be exploited, though

they cannot be used directly as judgments of absolute relevance (Cen et al., 2009).

Multiple issues are related to the storage component. A first issue concerns privacy since our approach aims at sharing information about searches that could reveal users' identities. This issue constitutes a recent subject of research (Navarro-Arribas et al., 2012; Poblete et al., 2010). A second issue concerns storage optimization. This issue raises many questions. First, it seems obvious that all the submitted queries do not need to be kept for future use. A strategy has to be defined on what types of queries should be stored and their storage durations. This may take inspiration from work on data storage (You et al., 2011). Second, different granularities of stored searches can be managed. A first option is to store past searches separately. Another option may consider a fusion of past searches corresponding to the same information need. A third option is to manage clusters of past searches.

Eventually, the update of the stored searches is a process to study. Thus, if past searches are stored separately, when a new user submits once more a query already stored, the returned documents judged relevant by the user have to be incorporated into the associated documents of the stored query. If a fusion approach is applied for storage, when a new submitted query is related to an already stored query, the new query has to be fused with the stored query and the associated result has to be updated according to the new user's judgments. If a clustering approach is applied for storage, when a new submitted query is related to an existing cluster, the new query and its result have to be included in the cluster.

On the other hand, the retrieval component aims at looking for stored searches with queries similar to a new submitted query and selecting documents from their associated past results, in addition to usual retrieval processes such as matching and ranking functions. Since this article focuses on this component, we detail the process executed when a new query is submitted. This process is the following:

- The new query q is compared with the queries representing the stored past searches.
- If stored searches are found as similar to the new query then their relevant documents constitute a set of possible candidates to compose the result for the new query q.
- A subset of documents is selected from the set of possible candidates to respond to the new query. Various approaches for selecting documents can be applied such as the two algorithms studied in this article.
- The past searches similar to the new search that were used to build the result answering the new query are then updated according to the user's judgments on this result.
- A systematic combination of traditional retrieval and retrieval using past searches might be applied.
- Else when no similar queries are found in the stored past searches, a traditional retrieval process is performed. The new query and its result are added to the set of stored past searches.

Different solutions can be applied at different steps of this retrieval process. Section 'Approach for reusing past queries' describes an approach based on the storage of past searches and the reuse of the most similar past query. Two algorithms that can be applied for selecting documents to build the result returned for a new submitted query are presented.

APPROACH FOR REUSING PAST QUERIES

This section presents an approach for reusing past queries that fits into the global process introduced in section 'Principles for collective reuse of past searches'. The basic idea of our approach is to incorporate to the system every query with its set of associated documents (query plus the list of documents returned as result for this query). The system uses thus not only a set of documents but also the queries executed by users (past queries) with their sets of associated documents. At the beginning, there are just

documents without queries. Every time that a query is submitted to the system, it is included to the system. When a new query is executed, first, it is checked and compared with the past queries, which are stored in the system with their documents. Then, from the most similar past query, relevant documents are obtained by applying our randomized algorithms. Our method consists of two parts. On one hand, submitted queries are stored with their documents. On the other hand, each new query is checked and compared with the past queries stored in the system. If there is a similar past query in the system, the relevant documents of this past query are selected by the algorithms. Relevant documents are thus used to respond to the new query.

Various probabilistic methods can be found in IR, in particular the Monte Carlo method (Burgin, 1999). Our probabilistic algorithms correspond to the Monte Carlo category. Monte Carlo algorithms supply an answer, which can be unsoundness (i.e., when the algorithm gives true, it could be false, or when the algorithm gives false, it could be true). When the algorithm gives true or false, and one of these answers is accurate, it is called true-biased. Otherwise, when both answers can be unsoundness, it is called two-sided errors. An example of true-biased Monte Carlo algorithm is when considering an unsorted array in which one wants to find a number, but reviewing only k elements of the array (k < N), to reduce the searching time. If the searched number is inside the k elements, the algorithm should return true (it is soundness). On the contrary, if the searched number is in the other portion of elements (N - k), the algorithm will respond false, even though the number is inside the array.

In our particular case, our algorithms are two-sided errors (i.e., both answers true or false can be inaccurate). The reason is that we are not sure about judgments of users with respect to whether a document is either relevant or non-relevant regarding the query. Nevertheless, we can speculate that whether the document position (in the list of documents) is at the top (closer to the query), the document has a higher probability to be relevant than a document that is at the bottom of the list.

The first algorithm (Algorithm 1) splits the list of retrieved documents in groups. The groups have different probabilities to have a *hit*, and at the same time, the elements of each group have different likelihoods. The first group, which is composed of the first elements appearing at the top of the list, has the greatest probability to get a *hit*. At the same time, the elements of this group have different likelihoods to have a *hit*. The first element of the group has a higher likelihood to have a *hit* than the last element of the group.

The second algorithm (Algorithm 2) is inspired from logistic distribution. The traditional function was modified for a polynomial of degree 1 to obtain a bound, which is used to calculate the likelihood considering a margin of error according to the position of an index in the result list.

Definitions 1: Let *DB* be an IR dataset composed of a set of documents *D*, a set of past queries *Q*, and a set of new queries *Q'* such that: a) all the documents are different; b) there are no common terms between past queries; c) there are no common terms between new queries.

Definitions 2: Let $V_N(q)$ be an ordered set (according to a similarity measure between documents and the query q such as the cosine measure) of *N* retrieved documents $d_i \in D$, given a query $q \in Q \bigcup Q'$.

Let $A(q)$ be the ordered subset of all the relevant documents retrieved for the query *q*.

Let $A'(q)$ be the ordered subset of all the irrelevant documents retrieved for the query *q* (*i.e.,* $A'(q) = V_N(q) \setminus A(q)$).

Definitions 3: Let $p_q = (q,\ V_N(q))$ be the search related to the query q, $\forall q \in Q \cup Q'$ (i.e., the pair composed of the query and its retrieved documents).

$P = \bigcup_{q \in Q} p_q$

is the set of all the searches stored in the system (see section 'Principles for collective reuse of past searches').

Definitions 4: Given a query $q' \in Q'$:

$$R_p\left(q'\right) = V_N\left(q\right), \text{such that } q \in Q \wedge p_q \in P \wedge sim\left(q', q\right) = \max_{\forall q_i \in Q \wedge p_{q_i} \in P}\left(sim\left(q', q_i\right)\right)$$

corresponds to the set of retrieved documents of the most similar past query, according to the similarity measure *sim* (e.g., cosine distance).

$\psi : R_p\left(q'\right) \rightarrow V_N\left(q'\right)$ is a function, which selects relevant documents in $R_p\left(q'\right)$ to define the retrieved documents for the new query $q' \in Q'$.

Definitions 5: Let *B* be a binary array of length *N*, and let:

$$\frac{a}{b}$$

be the proportion of values in *B* that are equal to 1 (true). This array is the basis for providing a level of general probability for all the documents. The probability of each document is computed according to the selected approach (i.e., Algorithm 1 or Algorithm 2, see Figures 1 and 2), according to its position in the ordered set $V_N\left(q\right)$.

Definitions 6: Let $M\left(N\right)$ be the upper approximation of $R_p\left(q'\right)$ length in power two. $M\left(N\right) = NG \times 2^{ne}$, where NG is the number of document groups, and 2^{ne} corresponds to the number of documents per group.

Considering *i* as the position of a document in $R_p\left(q'\right)$, such that the first element (*i* = 1) corresponds to the most similar document, then:

$$G\left(i\right) = \min_{x \in \mathbb{N} \wedge x \in \left[1, NG\right] \wedge i \leq x \times 2^{ne}}\left(x\right)$$

gives the group of the element *i*.

Let:

$$\varepsilon\left(i\right) = 1 - \left\{log_2\left(2^{M(N)} - \left(i mod\left(G\left(i\right)+1\right)\right)\right)\right\}$$

be the error assigned for the document at the position *i* in $R_p\left(q'\right)$.

Let:

$$K\left(i\right) = log_2\left(\frac{1}{\mathring{a}\left(i\right)}\right)$$

be the number of iterations on *B,* to assign the likelihood for the document at the position *i* in $R_p\left(q'\right)$.

Thus, $F\left(i\right) \rightarrow \left\{0,1\right\}$, is the probability function used by Algorithm 1 (see Figure 1), such that:

$$F\left(i\right) = \begin{cases} 1 : Pr_i\left(1\right) = \sum_{l=1}^{K(i)} \frac{2^{\left(M(N) - G(i)\right)}}{\left(2^{M(N)}\right)^l}, \\ 0 : Pr_i\left(0\right) = 1 - Pr_i\left(1\right) \end{cases}$$

where $Pr_i\left(1\right)$ is the likelihood of *hit* (1) for the element *i*, and $Pr_i\left(0\right)$ represents the probability of *miss* (0) for the element *i*.

Definitions 7: Let:

$$logit\left(i\right) = \log\left(\frac{p_1'}{1 - \left(\alpha + \frac{\beta}{i}\right)}\right)$$

Figure 1. Algorithm 1 for selection of documents in past results to build a new result

Algorithm 1
Input: q' is the new query, q is the most similar query with respect to q'
Ouput: VN' is the list of retrieved documents for the query q'.

```
1:Begin
2:  VN' ← ∅
3:  index.Inf ← 0
4:  index.Sup ← 0
5:  for (i = 1; i ≤ N; i++) do
6:    B[i] ← false    /* B is a Boolean array of length N. */
7:  end for           /* It provides the basic probability, true or false, see Definitions 5 */
8:  for (i = 1; i ≤ N/2; i++) do
9:    j ← randint(1, N)  /* generates an integer from 1 to N */
10:   B[j] ← true
11: end for
12: i ← 1
13: for (b = 1; b ≤ NG; b++) do
14: /* NG is the number of document groups in the result list given q' */
15:   for (c = 1; c < 2^ne; c++) do   /* 2^ne is the number of documents per group */
16:     E ← 1 − {log2(2^M(N) − c)}
17:     K ← ⌈log2(1/E)⌉    /* K gives the likelihood of documents according to */
18:                        /* their positions in VN, see Definitions 6          */
19:     for (l = 1; l ≤ K; l++) do
20:       if b = 1 then
21:         Index.Inf ← 1
22:         Index.Sup ← N
23:       else
24:         val ← randint(0, 1)
25:         if val = 0 then
26:           index.Inf ← 1
27:           index.Sup ← N/2^b
28:           else
29:           index.Inf ← N/2
30:           index.Sup ← N/2 + N/2^b
31:           end if
32:       end if
33:       indexF ← index.Inf + randint(1, index.Sup)
34:       if B[indexF] then
35:         if (VN(i) is in A) then   /* VN is the list of retrieved documents for q  */
36:                                   /* A is the sublist of relevant documents for q */
37:           Add VN(i) to VN'
38:           i ← i + 1
39:         end if
40:       else
41:         i ← i + 1
42:       end if
43:     end for
44:   end for
45: end for
46:End
```

Figure 2. Algorithm 2 for selection of documents in past results to build a new result

Algorithm 2
Input: q' is the new query, q is the most similar query with respect to q', $P'1$ is the initial probability
Output: VN' is the list of retrieved documents for the query q'

```
1:Begin
2:  VN' ← ∅
3:  for (i = 1; i ≤ N; i++) do
4:    B[i] ← false    /* B is a Boolean array of length N. */
5:  end for           /* It provides the basic probability, true or false, see Definitions 5 */
6:  for (i = 1; i ≤ N/2; i++) do
7:    j ← randint(1, N)    /* generates an integer from 1 to N */
8:    B[j] ← true
9:  end for
10: for (i = 1; i ≤ N; i++) do
                  log((P'1)/(1−(α+β/i)))
11:   E ← e                            ×γ      /* α, β, γ are real constants */
12:   if i = 1 then
13:     E ← 1 − P'1
14:     K' ← ⌈log2(1/E)⌉
15:   else
16:     K' ← [E]
17:   end if  /* K' gives the likelihood of documents according */
18:           /* to their positions in VN, see Definitions 7      */
19:   for (l = 1; l ≤ K'; l++) do
20:     indexF ← randint(1, N)
21:     if B[indexF] then
22:       if (VN[i] is in A) then   /* VN is the list of retrieved documents for q */
23:                                 /* A is the sublist of relevant documents for q */
24:         Add VN[i] to VN'
25:       end if
26:     end if
27:   end for
28: end for
29:End
```

be a bound used to calculate the number of iterations, according to the position i of the document in: $R_p\left(q'\right)$. $p_1^{'}$ corresponds to the initial probability and:

$$\alpha + \frac{\beta}{i}$$

is a polynomial of degree 1.

Let $K'\left(i\right) = e^{logit(i)} \times \gamma$ be the number of iterations on B for each $i > 1$. γ is a real number. For:

$$i = 1,\ K'\left(i\right) = \log_2\left(\frac{1}{1 - p_1^{'}}\right).$$

Thus, $F'\left(i\right)$ is the probability function used by Algorithm 2 (see Figure 2), such that:

$$F'\left(i\right) = \begin{cases} 1 : Pr_i^{'}\left(1\right) = \sum_{l=1}^{K'(i)} \frac{1}{2^l}, \\ 0 : Pr_i^{'}\left(0\right) = 1 - Pr_i^{'}\left(1\right) \end{cases}$$

where $Pr_i^{'}\left(1\right)$ is the likelihood of *hit* (1) for the element i using Algorithm 2, and $Pr_i^{'}\left(0\right)$ represents the probability of *miss* (0) for the element i.

It is important to point out that Definitions 1 to 5 are used for both algorithms. Algorithm 1 uses Definitions 6, while Algorithm 2 uses Definitions 7.

The key steps of Algorithms 1 and 2 are illustrated in the following examples. It is assumed that a set of past queries with their documents is available in the system. A new query *q'* is submitted to the system and a similar past query q is found. Algorithm 1 or Algorithm 2 is then applied.

In Algorithm 1, the list of documents *VN* retrieved for the most similar query *q* is splits in NG groups (e.g., $NG=4$), with the same quantity of documents ($2^{ne}=8$). The probability for each document is determined by two factors. First, it depends on the group the document belongs to, and second it depends on the relative position of the document in the group. For example, let us consider that the list of 30 retrieved documents for the most similar query is <d_1, d_2, …, d_9, d_{10}, …, d_{30}>. The document d_1 is at the first position in the first group, while the document d_9 is at the first position in the second group. For each document (Algorithm 1, line 15) of each group (Algorithm 1, line 13), the number of iterations *K* to assign the likelihood of each document is computed. For example, E = 0.05 for the documents d_1 and d_9 (Algorithm 1, line 16 and Definitions 6). Moreover, the number of iterations K (Algorithm 1, line 17 and Definitions 6) to find 1 inside B is the same. Nevertheless, the probability in the first group is bounded by: $\frac{2^4}{32}$ (Algorithm 1, lines 27 and 30). The probability for the document in position *i* to be added to the result *VN'* of the new query *q'*, considering the group *g* (from 1 to NG) it belongs to, is given by: $\sum_{l=1}^{K}\frac{1}{\left(2^g\right)^l}$ (Algorithm 1, loop For lines 19 to 43). Let us consider the parameters of the considered example as: N=30 (list length of retrieved document for *q*), NG = 4 (number of document groups), ne = 3 (2^{ne} documents per group) and $M(N)$ = 5 (see Definitions 6). For the document d_1 (first element i = 1 of the first group $G(1)$=1), the number of iterations $K\left(1\right)=5$ (Algorithm 1, lines 16 and 17). Then, since document d_1 belongs to the first group, its probability to be added to *VN'* (Algorithm 1, lines 34 to 37):

$$Pr_1\left(1\right)=\frac{1}{(2^1)^1}+\frac{1}{(2^1)^2}+\frac{1}{(2^1)^3}+\frac{1}{(2^1)^4}+\frac{1}{(2^1)^5}=\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+\frac{1}{32}=0.968$$

In a similar way, for the elements of the second group, the search to find a 1 inside the array *B*, is limited by the range 1 to: $\frac{N}{2}$ or $\frac{N}{2}+1$ to N (Algorithm 1, lines 26 to 30). Finally, the probability for the first document of the second group d_9 to be added to *VN'*:

$$Pr_9\left(1\right)=\frac{1}{(2^2)^1}+\frac{1}{(2^2)^2}+\frac{1}{(2^2)^3}+\frac{1}{(2^2)^4}+\frac{1}{(2^2)^5}=\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\frac{1}{256}+\frac{1}{1024}=0.333$$

Considering Algorithm 2, let us consider the following parameter values: $\alpha=$ **-4**, $\beta=$ **2.0**, $p_1^{'}=0.92$, and $\gamma=$ **10**. It is important to highlight that the middle of elements in *B* is valued to 1 and the rest of elements are valued to 0. For each document (Algorithm 2, line 10), the number of iterations *K'* to assign the likelihood of each document is computed (Algorithm 2, lines 11 to 16) depending on the position *i* of the document in the result list. The probability for the document in position *i* to be added to the result *VN'* of the new query *q'* is given by: $\sum_{l=1}^{K'}\frac{1}{2^l}$ (Algorithm 2, loop For lines 19 to 27). For the first document d_1 (i.e., $i=1$), $logit\left(1\right)=-1.181$ and $K'\left(1\right)=4$. Its probability to be added to *VN'* (Algorithm 2, lines 21 to 24)

$$Pr'_{01}(1) = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = 0.9375$$

In a similar way, for the 9th document d_9, $i = 9$, $logit(9) = -1.647$, and $K'(1) = 2$. Its probability to be added to *VN'* is thus given by:

$$Pr'_{09}(1) = \frac{1}{2} + \frac{1}{4} = 0.75$$

SIMULATION OF INFORMATION RETRIEVAL COLLECTIONS

A typical IR collection, following the Cranfield paradigm is composed of three elements: a set of documents, a set of queries, and query relevance judgments. Queries are usually defined by a list of terms. Relevance judgments are sets of document identifiers for each query, listing documents considered as relevant for the query (and sometimes listing documents not relevant for the query) (Voorhees & Harman, 2005). Documents and queries are relatively easy to find. Relevance judgments are usually generated by humans, and so more difficult to obtain since requiring significant effort and resources. Solutions to reduce the cost have been proposed (Sanderson, 2010). For example, (Alonso & Mizzaro, 2012) proposes a solution based on crowdsourcing. Other solutions adapt IR collections designed for a particular search task to evaluate other search tasks. For example, (Cetintas et al., 2011) adapted traditional collections with independent queries for approaches based on past queries to select resources in distributed IR.

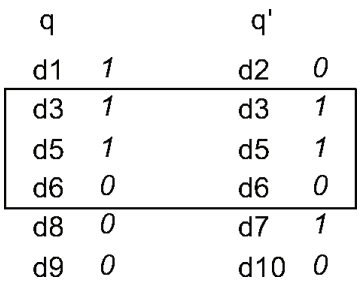
Simulation is seen as an interesting alternative for evaluation of IR approaches (Azzopardi et al., 2011). In this context, we propose a method to simulate an IR collection to evaluate approaches based on the use of similar past queries, based on prior work about IR dataset simulation (Gutiérrez-Soto & Hubert, 2013). This method includes simulation of documents and queries related to various topics, as well as users' judgments. Several parameters allow to vary topic distribution among documents and queries, and to vary relevance distribution in users' judgments. So, different simulated datasets can be built by varying these parameters.

Our method is separated into two steps. It relies on well-known laws of information science used in various research fields such as Digital Libraries and Information Retrieval (Chen & Leimkuhler, 1986; Sparck Jones & Willett, 1997; Schaer, 2013). The first step concerns the creation of terms, documents, and then queries. Heaps' law (Heaps, 1978), Zipf's law (Zipf, 1949), and exponential distribution are considered for this step. The second step concerns the generation of relevance judgments considering Bradford's law.

In our context of reusing past searches, an additional point concerns similar queries. Two similar queries are considered as referring to the same information need, as usually supposed in information retrieval. Consequently, a particular set of documents is relevant for two similar queries. Furthermore, there are other documents that are relevant for each of the two similar queries (i.e., some documents are relevant only for one of the two similar queries and other documents are relevant for both queries). For example, in Figure 3, documents d_3 and d_5 are relevant for both queries while d_1 is relevant for the query *q* only, and d_7 is a relevant document for *q'* only.

To simulate the aforementioned scenario about relevance judgments for two similar queries, three successive steps based on Zeta distribution are processed. Zeta distribution provides a discrete approximation of the Bradford's law. Let us consider a result list of documents retrieved for each query (e.g., in Figure 3, the list of documents $<d_1, d_3, d_5, d_6, d_8, d_9>$ is the result for query *q* and the list of documents $<d_2, d_3, d_5, d_6, d_7, d_{10}>$ is the result for query *q'*). The Bradford's law is applied to the sublist of documents common to both result lists (e.g., in Figure 3, the intersection of both result lists is the sublist $<d_3, d_5, d_6>$ and applying Zeta distribution, the documents d_3 and d_5 are designated relevant while d_6 is not relevant). Then, Zeta

Figure 3. Distribution of relevant and irrelevant documents for two similar queries (relevant = 1 and non relevant = 0)



distribution is applied to each document list of each query, maintaining the relevant documents in common previously determined (e.g., in Figure 1, d_1 is added to relevant documents for query *q*, while d_7 is added to relevant documents for query *q'*). The two queries share thus some relevant documents but have two different lists of relevant documents (i.e., each query has its own relevant documents). Therefore, when evaluating effectiveness of IR systems, for example according to P@10, precisions are different for the two similar queries with a given result list. Notice that this method does not favor any of the two queries (i.e., the new query and its corresponding past query) according to precision. Best precision is reached either for the past query or the new one.

Creation of Terms, Documents, and Queries

We consider documents and queries represented by sets of terms. These documents and queries are related to topics. As a first step, terms are generated from letters belonging to an alphabet such as the English alphabet. Each letter composing a term is chosen according to uniform distribution, each generated term is unique. Then, according to the Heaps' law a document with size $O(n)$ (where *n* is the number of terms) has a vocabulary (i.e., distinct terms) of size $O(n^{\beta})$ (where $0<\beta<1$), close to $O(\sqrt{n})$ (De Moura et al., 2000; Navarro et al., 2000). So, with *t* distinct generated terms we can create documents comprising around $O(t^2)$ terms. We assume that all the terms composing a document are representative. Consequently, usual processes such as removing stop words and stemming are not applicable. A second aspect concerns topic definition. Terms are split in different subsets, each subset corresponding to a particular topic. Then, documents are defined by selecting terms from topics. We assume that each document is related to one main topic and various minor topics. Usual distributions observed in IR, such as Zipf's distribution or exponential distribution, can be used to simulate term frequencies. By changing of distribution method and varying their coefficients, different collections of documents can be built, representing different scenarios.

Then, past queries are created from documents. Like for documents, terms are chosen under uniform distribution to build past queries. Past queries are defined in such a way that intersections between past queries are empty. Finally, new queries are built from past queries. For each past query a new query is defined, either by changing or adding a term. The most similar past query corresponding to a new query is thus always the one used to define it.

Generation of Relevance Judgments

We base on the Bradford's law to simulate relevance judgments for a given query, traditionally provided by users. The Bradford's law states that most relevant papers are in few journals while other relevant papers are spread on a high

quantity of other journals (Garfield, 1980). By analogy, we consider a result list of documents retrieved by an IR system as a succession of document groups. Each group is seen like a journal in the Bradford's law. We assume that most of the relevant documents are in the first groups (i.e., top ranked documents), while the remaining relevant documents are spread on the next groups.

EMPIRICAL RESULTS

Experimental Environment

Experimental environment was set as follows. The English alphabet was used to create terms, and consequently topics, documents, and queries. The length of a term | t |, was between 3 and 7. Uniform distribution was used to establish the length. The number of terms | T | was 700 in each experiment. The number of terms for each document was between 15 and 30. According to Heaps' law, it was possible to represent documents between 300 to 900 words in our case. The number of topics used in each experiment was 7. Each topic was formed by 100 terms. When building a document, term frequencies were defined using either exponential distribution or Zipf distribution. Whereby, most words, which compose a document, were chosen from a specific topic. Document numbers used in each experiment were 700, 1400, 2100, 2800, and 3500. In addition, terms for a query were between 3 and 8. Both terms and documents were chosen using uniform distribution to build the past queries. We built a first set of 15 queries. Then, from this first set of queries, 15 new queries were built. Each query was used two ways: one, as new query; two, as past query for a new query. Consequently, each experiment used 30 queries.

Simulations of user judgments were carried out under Zeta Distribution. Zeta distributions with parameters 2, 3, and 4 have been applied to the 30 most similar documents with respect to the queries.

Experimental Results

In this section, three experiments are studied. Both, in the first and second experiments, exponential distributions were used to build the collection of documents *D*. In the third experiment, Zipf distribution was applied to build *D*. Additionally, we applied the Student's Paired t-Test (paired samples) over each P@10 (i.e., the precision at ten retrieved documents that is the fraction of the ten top ranked retrieved documents that are relevant) comparing our approach (i.e., each of our algorithms) with the traditional retrieval (i.e., cosine) it reuses past results, for each set of documents (700, 1400, 2100, 2800, and 3500). Results are shown according to average P@10 over all the queries. We applied Algorithm 2 with the parameters: $\alpha =$ **-4**, $\beta =$ **2.0**, $p_1^{'} = 0.92$, and $\gamma = 10$.

1. **Experiment 1:** Exponential distribution (with parameter $\theta = 1.0$) was used to build *D*. Zeta distributions were applied to determine the list of relevant documents for each query: with parameter $S=2$ for scenario *a*, $S=3$ for scenario *b*, and $S=4$ for scenario *c*. The average results for P@10 are displayed in Figure 4.

Algorithm 1 had an average of 5.7 queries that were not improved applying our approach (3.8 queries for scenario *a*, 7.4 queries for scenario *b*, and 5.8 for scenario *c*). An average of 22.6 queries led to result improvements (25.8 queries for scenario *a*, 21 queries for scenario *b*, and 21.1 for scenario *c*). The highest p-value was lower than 1.0 E-6 applying the Paired t-test.

Algorithm 2 presented an average of 10.6 queries that were not improved (10.6 queries for scenarios *a*, *b*, and *c*) and an average of 18.3 queries were improved (19.4 queries for scenario *a*, 18.4 queries for scenario *b*, and 17 queries for scenario *c*) using this algorithm. The highest p-value for the algorithm 2 was 1.71 E-5 applying the Paired t-test.

Figure 4. Experiment 1, evaluation results of the three approaches (i.e., Algorithm 1, Algorithm 2, and Cosine) according to average P@10 (over 30 queries) with a collection D based on exponential distribution using θ = 1.0 and a) Zeta distribution (with parameter S = 2) for relevant documents, b) Zeta distribution (with parameter S = 3) for relevant documents, and c) Zeta distribution (with parameter S = 4) for relevant documents.

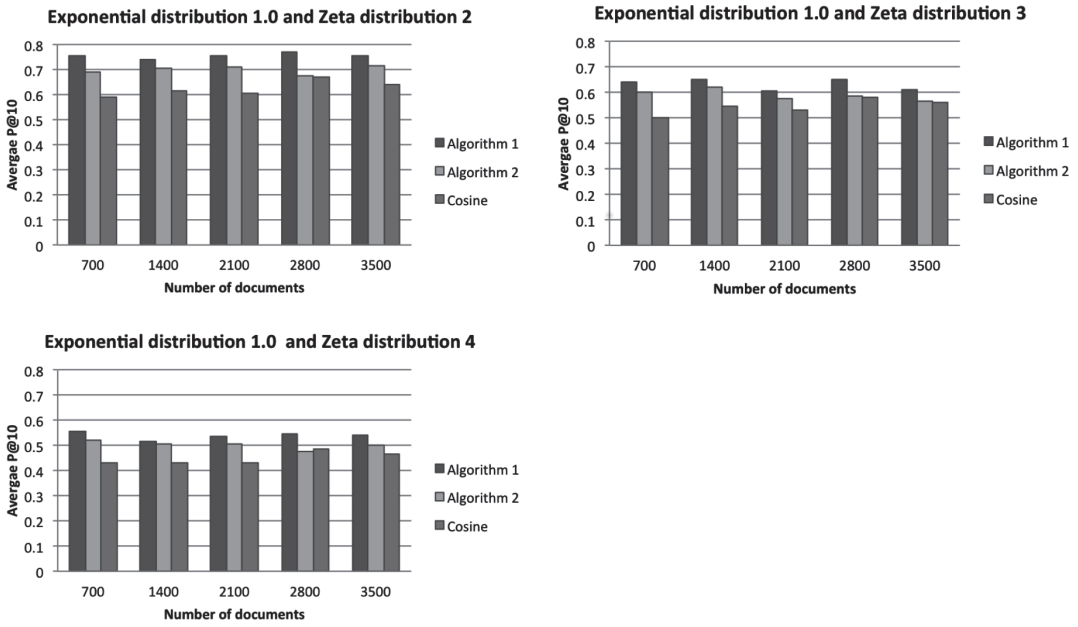


Figure 5. Experiment 2, evaluation results of the three approaches (i.e., Algorithm 1, Algorithm 2, and Cosine) according to average P@10 (over 30 queries) with a collection D based on exponential distribution using θ = 1.5 and a) Zeta distribution (with parameter S = 2) for relevant documents, b) Zeta distribution (with parameter S = 3) for relevant documents, and c) Zeta distribution (with parameter S = 4) for relevant documents.

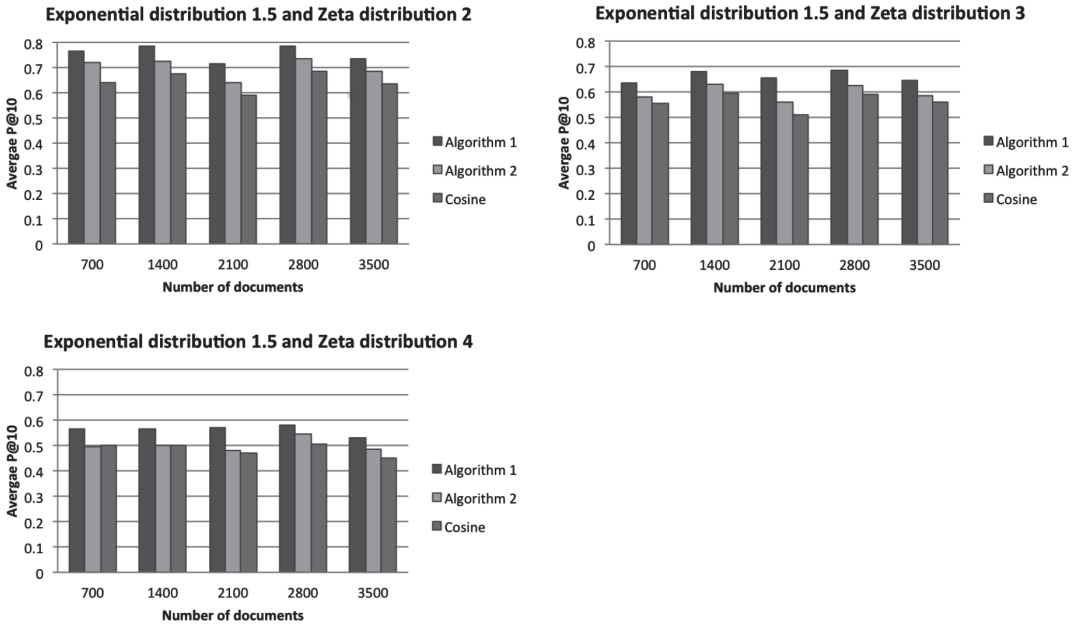
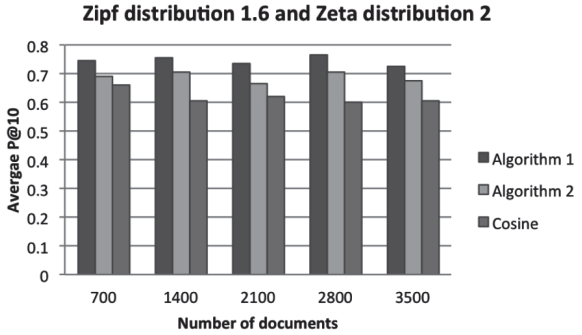
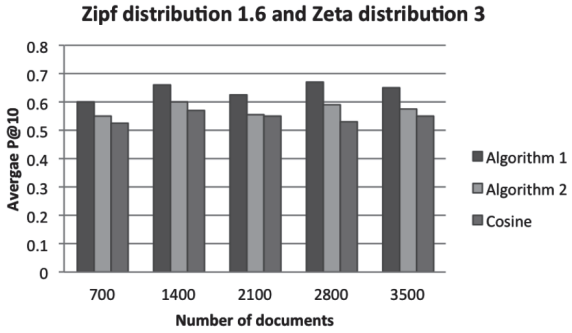
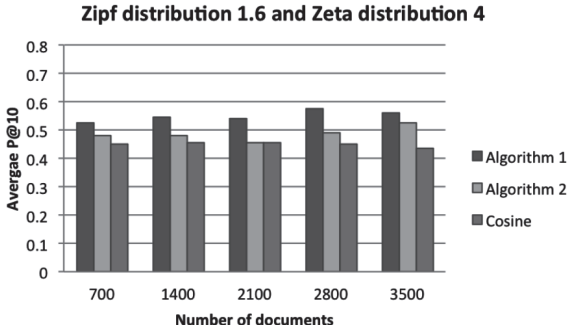


Figure 6. Experiment 3, evaluation results of the three approaches (i.e., Algorithm 1, Algorithm 2, and Cosine) according to average P@10 (over 30 queries) with a collection D on Zipf distribution using λ = 1.6 and a) Zeta distribution (with parameter S = 2) for relevant documents, b) Zeta distribution (with parameter S = 3) for relevant documents, and c) Zeta distribution (with parameter S = 4) for relevant documents.







2. **Experiment 2:** In this experiment, exponential distribution (with parameter θ = 1.5) was applied to build the collection *D*. Zeta distributions were applied to determine the list of relevant documents for each query: with parameter $S=2$ for scenario *a*, $S=3$ for scenario *b*, and $S=4$ for scenario *c*. The average results for P@10 are displayed in Figure 5.

Algorithm 1 had an average of 5.7 queries that were not improved applying our approach (4 for scenario *a*, 5.4 for scenario *b*, and 8 for scenario *c*). An average of 23 queries led to result improvements (25.6 queries for scenario *a*, 24 queries for scenario *b*, and 19.4 for scenario *c*). The highest p-value was lower than 1.0 E-6 applying the Paired t-test.

Algorithm 2 presented an average of 10.6 queries that were not improved (10.2 queries for scenario *a*, 11.4 for *b*, and 10.2 for *c*) and an average of 18.6 queries were improved (19.6 for scenario *a*, 18.6 for *b*, and 17.8 for *c*) using this algorithm. The highest p-value for the algorithm 2 was 0.028 applying the Paired t-test.

3. **Experiment 3:** In this experiment, Zipf distribution (with parameter λ = 1.6) was applied to build *D*. Zeta distributions were applied to determine the list of relevant documents for each query: with parameter $S=2$ for scenario *a*, $S=3$ for scenario *b*, and $S=4$ for scenario *c*. The average results for P@10 are shown in Figure 6.

The average number of queries, where Algorithm 1 did not improve results using past queries, was 4.8 (4 for scenario *a*, 6.8 for *b*, and 3.6 for *c*). The average number of queries where our approach improved results was 24.1 (24.8 for scenario *a*, 22.4 for *b*, and 25.2 for *c*). The highest p-value was lower than 1.0 E-6 applying the Paired t-test.

Algorithm 2 had an average of 10.7 queries that were not improved (9.8 for scenario *a*, 11.2 for *b*, and 11 for *c*) and an average of 18.3 queries were improved using this algorithm (19.6 for scenario *a*, 17.4 for *b*, and 17.8 for *c*). The highest p-value was 0.019 for the Paired t-test.

Discussion

On one hand, our main argument, by which both Zipf and exponential distributions were used, is because in documents, it is feasible to find terms not only about a particular subject but also other subjects. In our experimental environment, Zipf distribution was used with value 1.6 since accepted ranges are usually between 1.4 and 1.8.

In addition, to analyze if the function simulating relevance judgments has implications on results according to P@10, Zeta distributions with different parameter values (i.e., $S = 2$, 3, and 4) were applied. According to Figures 4 to 6, every time the parameter S was incremented, averages of P@10 decreased for both approaches, i.e., Algorithms 1 and 2 (our approach) and Cosine (using traditional IR). This can be noticed in every configuration used to build the document collections.

It should be noticed that Algorithm 1 provided the best average P@10 in every experiment, with p-values always supporting statistically significant results.

On the other hand, queries considered in this paper are defined by a set of terms only (i.e., without operator). This is the common type of query in general search engines (White & Morris, 2007). More specific information retrieval systems processing more complex queries (for example, conjunctive queries) would require defining a more sophisticated similarity measure between queries.

In addition, our approach is not based on a particular IR model since it reuses previous results returned by a given IR system. Thus, our approach depends on the initial IR system used. For this reason, it makes sense to compare our approach with the approach it reuses past results. The experiments reported in this article are based on a simple cosine-based IR system. Complementary experiments based on more sophisticated approaches could be carried out to observe if there are similar differences between our approach and the (more sophisticated) approach it reuses previous results.

CONCLUSION AND FUTURE WORK

In information retrieval systems, the retrieval process is usually entirely accomplished the same way from the beginning. In addition, all the information related to a search is not saved after the system has returned the results to the user. However, past searches are valuable information. When considered, past searches are mainly used for result personalization in an individual way, i.e., past searches of a given user are used to rerank the results for a new query submitted he/she submits, usually a repeated query.

Contrary to most of the approaches using past searching in an individual way, we presented in this paper an approach for capitalizing on past searches collectively. The approach assumes that a user may benefit from experiences of other users sharing the same information need through their past searches. The approach relies first on a storage process that was not detailed since out of the scope of this paper. We detailed in this paper the retrieval process based on past queries, especially regarding the selection of documents in past results to build results for new queries. We presented two probabilistic algorithms of the Monte Carlo category to select documents from past results. These algorithms are quite easy to implement and do not require learning.

A series of experiments were carried out to validate the interest of our approach with multiple objectives. First, they provided a comparison of our approach based on the reuse of past queries with the baseline approach of information retrieval it reuses past results. Second, they provided a study of two algorithms that can be used in our approach for the construction of a new result list from past

results. Eventually, they used different datasets to evaluate the behavior of our approach in different environments. Evaluations were made according to effectiveness (i.e., result accuracy) by measuring precision. Measuring precision at fixed ranking is a common option when considering that users choose only to examine a fixed number of retrieved results, and P@10 (i.e., the fraction of relevant documents in the ten top-ranked retrieved documents) is the commonest (Sanderson, 2010).

Faced with the difficulty to find IR collections suitable for evaluation of approaches based on past queries we opted for simulation to address this issue. We defined a method to simulate different IR collections to evaluate the performances of our approach based on the use of similar past queries. Simulation included document collections, queries, as well as users' judgments, and followed the Cranfield paradigm, well established in the IR domain. In this context, evaluation results show encouraging performances of our approach.

Since we are at the beginning of our approach development, there are many lines for future work. A first work will be to carry out experiments on a more traditional IR collection to evaluate the differences with our simulated scenarios. For this, we have to find an existing collection that could be adapted for our purpose. A second line concerns the selection of documents for new queries from multiple past queries. We are working on an approach based on clustering. Clusters of similar queries will be used instead of directly using queries. New submitted queries will be compared to centroids of clusters and documents for new queries will be selected among the documents associated to the queries of the chosen cluster. Finally, future work will concern the storage process, notably regarding storage optimization. Many questions are raised, related to the types of queries to keep and their storage time, related to information to store, and related to the updates of the stored past results according to users' interactions.

REFERENCES

Alonso, O., & Mizzaro, S. (2012). Using crowdsourcing for TREC relevance assessment. [Elsevier.]. *Information Processing & Management*, *48*(6), 1053–1066. doi:10.1016/j.ipm.2012.01.004

Azzopardi, L., de Rijke, M., & Balog, K. (2007). Building simulated queries for known-item topics: an analysis using six european languages. In *ACM International Conference, SIGIR'07* (pp. 455–462). ACM. doi:10.1145/1277741.1277820

Azzopardi, L., Järvelin, K., Kamps, J., & Smucker, M. D. (2011). Report on the SIGIR 2010 workshop on the simulation of interaction. *SIGIR Forum, 44*(2), 35–47.

Baeza-Yates, R., Castillo, C., Marin, M., & Rodriguez, A. (2005). Crawling a country: better strategies than breadth-first for web page ordering. In *Special interest tracks and posters of the ACM International Conference WWW '05* (pp. 864–872). ACM.

Burgin, R. (1999). The Monte Carlo method and the evaluation of retrieval system performance. [Wiley.]. *Journal of the American Society for Information Science*, *50*(2), 181–191. doi:10.1002/(SICI)1097-4571(1999)50:2<181::AID-ASI8>3.0.CO;2-9

Cen, R., Liu, Y., Zhang, M., Zhou, B., Ru, L., & Ma, S. (2009). Exploring relevance for clicks. In *ACM International Conference CIKM'09* (pp. 1847–1850). ACM.

Cetintas, S., Si, L., & Yuan, H. (2011). *Using past queries for resource selection in distributed information retrieval* (Tech. Rep. No. 1743). Purdue University: Department of Computer Science. Retrieved January 30, 2015, from http://docs.lib.purdue.edu/cstech/1743

Chen, Y.-S., & Leimkuhler, F. F. (1986). A relationship between Lotka's Law, Bradford's Law, and Zipf's Law. [Wiley.]. *Journal of the American Society for Information Science*, *37*(5), 307–314. doi:10.1002/(SICI)1097-4571(198609)37:5<307::AID-ASI5>3.0.CO;2-8

Clough, P., & Sanderson, M. (2013). Evaluating the performance of information retrieval systems using test collections. *Information Research, 18*(2). Retrieved January 30, 2015, from http://InformationR.net/ir/18-2/paper582.html

Cui, H., Wen, J.-R., Nie, J.-Y., & Ma, W.-Y. (2003). Query expansion by mining user logs. *IEEE Transactions on Knowledge and Data Engineering*, *15*(4), 829–839. doi:10.1109/TKDE.2003.1209002

Dang, V., & Croft, B. W. (2010). Query reformulation using anchor text. In *ACM International Conference WSDM '10* (pp. 41–50). ACM.

De Moura, E. S., Navarro, G., Ziviani, N., & Baeza-Yates, R. (2000). Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, *18*(2), 113–139. doi:10.1145/348751.348754

Fitzpatrick, L., & Dent, M. (1997). Automatic feedback using past queries: Social searching? In *ACM International Conference SIGIR'97* (pp. 306–313). ACM. doi:10.1145/258525.258597

Garfield, E. (1980). Bradford's Law and Related Statistical Patterns. *Essays of an Information Scientist, 4*(19), 476–483. Retrieved January 30, 2015, from http://www.garfield.library.upenn.edu/essays/v4p476y1979-80.pdf

Gayo-Avello, D. (2009). A survey on session detection methods in query logs and a proposal for future evaluation [Elsevier.]. *Information Sciences*, *179*(12), 1822–1843. doi:10.1016/j.ins.2009.01.026

Ghorab, M. R., Zhou, D., O'Connor, A., & Wade, V. (2013). Personalised Information Retrieval: Survey and classification [Springer.]. *User Modeling and User-Adapted Interaction*, *23*(4), 381–443. doi:10.1007/s11257-012-9124-1

Gutiérrez-Soto, C., & Hubert, G. (2013). Evaluating the Interest of Revamping Past Search Results. In *International Conference DEXA 2013* [Springer.]. *Proceedings Part II, LNCS*, *8056*, 73–80.

Heaps, H. S. (1978). *Information Retrieval: Computational and Theoretical Aspects*. Academic Press.

Hust, A. (2004). Introducing Query Expansion Methods for Collaborative Information Retrieval. *Adaptive READ Research Proje*ct [). Springer.]. *LNCS*, *2956*, 252–280.

Huurnink, B., Hofmann, K., De Rijke, M., & Bron, M. (2010). Validating query simulators: an experiment using commercial searches and purchases. In *International Conference CLEF'10* (pp. 40–51). Springer. doi:10.1007/978-3-642-15998-5_6

Jansen, B. J., Spink, A., & Saracevic, T. (2000). Real life, real users, and real needs: A study and analysis of user queries on the web. [Elsevier.]. *Information Processing & Management*, *36*(2), 207–227. doi:10.1016/S0306-4573(99)00056-4

Kanoulas, E., Carterette, B., Clough, P., & Sanderson, M. (2010). Overview of the TREC 2010 Session Track. In *19th TREC Conference*. Retrieved January 30, 2015, from: http://trec.nist.gov/pubs/trec19/papers/SESSION.OVERVIEW.2010.pdf

Marin, M., Gil-Costa, V., Bonacic, C., Baeza-Yates, R., & Scherson, I. D. (2010). Sync/async parallel search for the efficient design and construction of web search engines. *Parallel Computing*, *36*(4), 153–168. doi:10.1016/j.parco.2010.02.001

Navarro, G., De Moura, E. S., Neubert, M., Ziviani, N., & Baeza-Yates, R. (2000). Adding compression to block addressing inverted indexes. [Springer.]. *Information Retrieval*, *3*(1), 49–77. doi:10.1023/A:1009934302807

Navarro-Arribas, G., Torra, V., Erola, A., & Castellí-Roca, J. (2012). User k-anonymity for privacy preserving data mining of query logs. [Elsevier.]. *Information Processing & Management*, *48*(3), 476–487. doi:10.1016/j.ipm.2011.01.004

Poblete, B., Spiliopoulou, M., & Baeza-Yates, R. (2010). Privacy-preserving query log mining for business confidentiality protection. *ACM Transactions on the Web, 4*(3).

Potey, M., Patel, D., & Sinha, P. (2013). A survey of query log processing techniques and evaluation of web query intent identification. In *International Conference IACC* (pp. 1330–1335). IEEE Computer Society. doi:10.1109/IAdCC.2013.6514421

Sanderson, M. (2010). Test Collection Based Evaluation of Information Retrieval Systems. *Foundations and Trends in Information Retrieval*, *4*(4), 247–375. doi:10.1561/1500000009

Schaer, P. (2013). Applied Informetrics for Digital Libraries: An Overview of Foundations, Problems and Current Approaches. *Historical Social Research (Köln)*, *38*(3), 267–281.

Shen, X., Tan, B., & Zhai, C. (2005). Context-sensitive information retrieval using implicit feedback. In *ACM International Conference SIGIR'05* (pp. 43–50). ACM.

Shen, X., & Zhai, C. X. (2003). Exploiting query history for document ranking in interactive information retrieval. In *ACM International Conference SIGIR'03* (pp. 377–378). ACM. doi:10.1145/860500.860509

Song, S.-K., & Myaeng, S. H. (2012). A novel term weighting scheme based on discrimination power obtained from past retrieval results. [Elsevier.]. *Information Processing & Management*, *48*(5), 919–930. doi:10.1016/j.ipm.2012.03.004

Sparck Jones, K., & Willett, P. (1997). *Readings in Information Retrieval*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Steichen, B., Ashman, H., & Wade, V. (2012). A comparative survey of personalised information retrieval and adaptive hypermedia techniques. [Elsevier.]. *Information Processing & Management*, *48*(4), 698–724. doi:10.1016/j.ipm.2011.12.004

Tague, J. M., & Nelson, M. J. (1981). Simulation of user judgments in biblio- graphic retrieval systems. In *ACM International Conference SIGIR'81* (pp. 66–71). ACM.

Teevan, J. E., Adar, E. R., Jones, R., & Potts, M. A. S. (2007). Information re-retrieval: repeat queries in yahoo's logs. In *ACM International Conference SIGIR'07* (pp. 151–158). ACM.

Voorhees, E. M., & Harman, D. K. (2005). *TREC: Experiment and Evaluation in Information Retrieval*. Cambridge, MA, USA: MIT Press.

White, R. W., & Morris, D. (2007). Investigating the querying and browsing behavior of advanced search engine users. In *ACM International Conference SIGIR '07* (pp. 255–262). ACM. doi:10.1145/1277741.1277787

You, L. L., Pollack, K. T., Long, D. D. E., & Gopinath, K. (2011). *PRESIDIO: A Framework for Efficient Archival Data Storage. ACM Transactions on Storage, 7(2)*. ACM.

Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.