Randomized Algorithm for Information Retrieval Using Past Search Results

Claudio Gutiérrez-Soto, Gilles Hubert

Randomized Algorithm for Information Retrieval Using Past Search Results

Claudio Gutiérrez-Soto
Departamento de Sistemas de Información
Universidad del Bío-Bío, Chile
and
University of Toulouse, IRIT UMR 5505 CNRS, France
Claudio-Orlando.Gutierrez-Soto@irit.fr

Gilles Hubert
University of Toulouse
IRIT UMR 5505 CNRS
118 route de Narbonne
F-31062 Toulouse cedex 9, France
Gilles.Hubert@irit.fr

***Abstract*—In Information Retrieval, past searches are a source of useful information for new searches. This paper presents an approach for reusing past queries submitted to an information retrieval system and their returned results to build the result list for a new submitted query. This approach is based on a Monte Carlo algorithm to select past search results to answer the new query. The proposed algorithm is easy to implement and does not require learning. First experiments were carried out to evaluate the proposed algorithm. These experiments used a simulated dataset (i.e., document collections, queries and judgments of users are simulated). The proposed approach was compared with a traditional approach of information retrieval, showing better precision for our proposed approach.**

***Keywords*—Reusing past queries, probabilistic algorithm, IR.**

I. Introduction

A large assortment of contributions in information retrieval (IR), which address different perspectives such as matching functions, indexing, formal models and relevance feedback can be found in the literature. Nonetheless, few approximations take advantage of previously performed searches, in particular those of other users. Past searches can be a useful source of information for new searches. In many cases, a user may benefit from the search experiences carried out previously by a group of users with about the same information need.

This paper is interested in capitalizing on past queries submitted to an information retrieval system, to improve the result returned for a new query. What we propose in this paper is an approach for reusing past queries to respond to a new submitted query. The approach relies on two processes: a storage process and a retrieval process. This retrieval is based on a probabilistic approach.

Probabilistic methods are present in the literature of IR. Two threads are easily identifiable, i.e., learning techniques and optimization. Learning techniques involve Bayesian Networks and variants, to name just a few. Nevertheless, the scope of applicability of a learning algorithm, the achieved success in different environments as well as the lack of formal specific model are difficult features to address [1]. Optimization is related with Genetic Algorithms (GA), among others. However, with GA fitness evaluations are usually costly [2]. Furthermore, the same algorithm applied to different datasets can involve convergence times totally different [3].

Our approach involves a Monte Carlo algorithm, which is quite simple to implement, and does not require extra time to learn. The algorithm selects, in response to a new query, relevant documents from the most similar past search. The algorithm provides highest likelihood for documents, which appear at the top of list. This likelihood is assigned in a decreasing way according to the positions of documents in the result list of the past query.

In order to prove that our algorithm can improve precision (in particular P@10 precision (in this case, precision corresponds to the number of relevant documents retrieved in the first ten positions, which is divided by ten)) for new queries using past query results, a wide range of experiments have been carried out with comparison with a traditional retrieval. Experiments have been validated by applying the Student's Paired t-Test to support differences.

Ad-hoc collections for approaches based on similar past queries are difficult to find, because construction of these collections implies high cost not only in time but also in effort. Moreover, solutions to reduce the cost have been proposed in [4], [5]. Traditional collections of IR have been adapted to evaluate approaches based on past queries that respond to other issues [6]. In addition, in the context of the Web, it is possible to find approximations relying on historical searches. In this case, query logs are used as sources of information about users such as user clicks. Nevertheless, user clicks cannot be used directly as judgments of absolute relevance [7].

An alternative for the evaluation of information retrieval approaches corresponds to simulation [8]. Thus, we defined a method to simulate an environment allowing us to evaluate the performances of our algorithm relying on similar past queries.

This paper is organized as follows. Section II present the overall retrieval process capitalizing on past queries. In section III, related works on past searches, randomized algorithms, and simulation in IR context are presented. In section IV, our approach using past queries relying on a Monte Carlo algorithm is described, using mathematical definitions. In section V, we present our approach to simulate an IR collection, in the context of past search results. In section VI, empirical results are described to validate our approach and compare it with a traditional retrieval approach. Finally, the conclusions are presented in section VII.

II. Information retrieval reusing past queries

Most information retrieval systems do not differentiate users submitting queries. A given query leads to the same result list, except when a personalization process is applied to the result list. The possible personalization relies on user preferences explicitly given by users or deduced from the interactions of users during their previous searches. Past queries submitted by other users in the past constitute a useful source of information for new searches. Nevertheless, few systems exploit this source of information. Users may deplore the lack of memory shown by most information retrieval systems [9]. Many cases may benefit from past search exploitation. For example, a user can benefit from the search experiences carried out by a group of users and vice versa.

This paper aims to add a collaborative dimension in the information retrieval process by capitalizing on similar searches performed by other users. The approach relies on two separated processes: a storage process and a retrieval process.

On one hand, the storage process aims at recording the submitted queries and their computed results. The stored results comprise identifiers of documents 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 [10]. In the absence of explicit relevance judgments, user clicks could be exploited, although they cannot be used directly as judgments of absolute relevance [7]. When the same query is submitted several times by various users, all the relevance judgments on the documents constituting the successive results have to be combined. This process builds a set of query profiles. Each query profile is composed of the query expression (i.e., terms) and the identifiers of the documents with relevance judgments.

On the other hand, the retrieval process relies on the following steps:

- Each new query q is compared with the past queries of the stored set of query profiles.
- If there are past queries quite similar in the system then the relevant documents in their profiles 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.
- If no similar query is found in the set of past query profiles a traditional retrieval process is performed. A systematic combination of traditional retrieval and retrieval using past queries might be applied.
- the stored set of query profiles is then updated according to relevance judgments on the result returned for the new query q. If the new query is not present in the set of past query profile then the profile of the new query is added to the set of past queries. If the new query is present in the set of past query profile then its profile is updated according to the new relevance judgments.

Section IV describes a feasible approach of such retrieval process.

III. Related Work

A. Past Searches in IR

A few contributions in IR deal with the use of past queries. In [11], aiming to increase the average precision, two strategies are carried out. First, the new result is obtained by combining results of previous queries. Second, the new result is provided by merging query models, where an estimator of a word according to the context-based query model is used. An extension of the previous work is presented in [12]. The authors aim to improve the precision, specifically through models based on implicit feedback information, which are provided by queries and clickthrough history on an active session. This implies that the set of queries is executed by the same user, in the same context. The Kullback-Leibler divergence method is used to yield a score to a document. Bayesian interpolation is used to rerank any document that has not been seen by a user. It should be noted that the TREC collections used here were modified to evaluate this approximation. In [6], a distributed approach relying on the use of past queries is presented. Like previous work, a set of similar queries have been simulated from a query collection. With the initial used collection, it was not possible to evaluate schemes under similar queries. It is crucial to emphasize that relevance judgments of past queries were omitted. Hence, it is difficult to appreciate the real performance of the use of similar past queries.

In a similar way, some approaches that are related with the use of past queries can be found in the context of the Web. In [13], the authors provide an automatic method to produce suggestions, which are linked to previously submitted queries. To that end, an algorithm of association rules was applied on 95 most popular queries present in a log file. Nevertheless, the percentage of records corresponding to those queries over 2.3 millions of records in the log file was omitted. As a consequence, it is difficult to calculate the impact of this approximation. Furthermore, in [14], the authors point out that it is not easy to estimate the real impact of approximation based on association rules using log files. It is tricky to determine the successive queries, which belong to the same session (for the same user). Therefore, queries that could be related cannot be assigned to the same user. In [15], an architecture of collaborative search, which takes advantage of repeated queries is proposed. Search results are customized for different communities of users. Final results are highly related to communities with the same background. Another approach that relies on repeated queries is exposed in [16]. This research aims at identifying identical queries executed in the same trace. In [17], two contributions that benefit from repeated queries are presented. One is focused on efficiency in execution time, and the second relies on repetitive document access by the search engines.

In summary, approximations relying on past queries in the context of the Web, are focused rather on repeated queries in log files than similar queries. Moreover, although log collections can provide a framework to evaluate the use of past queries, their use is complex because there are no relevance judgments of documents for a given query. Thus, nowadays

it is difficult to find collections including documents, queries, and relevance judgments to evaluate in an objective way the utility about the use of similar past query results. This can be due to different factors: the high cost of construction of ad-hoc environments, and the time involved to evaluate not only similarities between queries but also the relevance of documents for similar queries.

B. Simulation in IR evaluation

A wide range of approaches, which involve the use of simulation can be found since 1980s. In [8], [18], simulation to evaluate IR systems is presented as further avenues of research. Simulation in IR can concern not only document collections but also query sets with their relevance judgments, which can be provided without user interaction [19]. In [20], an algorithm was developed to simulate relevance judgments of users. In this work, the real precision was compared to the simulated precision. In [21], queries to find known-item were simulated. The identification and generation of methods were carried out to obtain simulated topics as similar as possible to real topics. Other researches to simulate user interactions through click log, queries, and preferences of users have been built [22], [19]. Moreover, nowadays with the exponential growth of the Web, simulation gives interesting approximations of performance on the Web [23], [24].

C. Randomized Algorithms

Regarding the usage of probabilistic algorithms in IR, several contributions can be found in the literature. Most of them rely on learning techniques or optimization. Among existing optimization techniques, we can find Genetic Algorithms (GA). In [25], a new fitness function based on the formula proposed by [26] (where the term weights for documents and queries are the product between the term frequency multiplied by an inverse collection frequency factor) is given. In this work, both vectors of documents and queries are normalized using the formula. Empirical results showed a more effective way using this approach than a traditional retrieval (i.e., using cosine distance). In [27], an algorithm called *probfuse*, whose purpose is to mix results from several IR algorithms, is compared to the widely used CombMNZ algorithm [28]. Final results show a better performance of *probfuse*. In [29], GA and Genetic Programming, to study the impact in three areas such as the use of documents to improve precision, ranking methods, and studies about the combination of the previous two areas on MAP, were presented.

Among approximations relying on learning techniques, Bayesian Networks and their variants can be found in the IR context. In [30], a probabilistic version of TFIDF algorithm called PrTFIDF is shown. PrTFIDF gives a new sight about vector space model, where a descriptor for every document, the theorem of total probability and the Bayes' theorem are used. Final results on Usenet articles provide a better performance than TFIDF, on six classification tasks. Another approach aiming to give an unsupervised relevance to classify documents according to the query or topic using Poisson distribution, is exposed in [31]. However, as we mentioned in the introduction, these approaches are not always simple to implement, and furthermore, require high execution times.

In contrast to GA and Bayesian Networks, Las Vegas and Monte Carlo algorithms are applied when the problem is hard to solve like NP problems (i.e., time is a relevant factor in order to give the answer) or algorithm input is non-deterministic. An example of non-deterministic problems is a card game. In such a game, someone tries to guess a card. In every round the probability to guess is increased. Finally, in the 51st round, both cards have achieved the maximum percent to be chosen. In this example, it is always possible to find a mistake probability (except for the last card).

Las Vegas algorithms provide an answer either *true* or *false*, and this answer is always correct. To illustrate it, an example is provided. Given an unsorted array, one wants to find a number. Here, the algorithm provides a random position (from *1* to *N*), and the searched number is compared with the number, which is at the random position. If both numbers are equal, the algorithm returns *true*. Otherwise, the number which is in the random position is swapped for the number at first position. Now, the algorithm can choose a random position (from *2* to *N*). Again the searched number is compared with the number, which is at the random position. If the searched number is different to the number at the random position, the number at the random position is swapped for the number at second position, and so on. Finally, the number can always be found because it is inside the array.

On the other hand, if the searched number is not inside the array, the algorithm should respond *false*. In both cases (*true* or *false*), the answer is always correct.

Unlike Las Vegas algorithms, Monte Carlo algorithms give an answer, which can be incorrect (it means, when algorithm gives *true*, it could be *false*, or when algorithm gives *false*, it could be *true*). When the algorithm gives *true* or *false*, and one of these answers is correct, it is called true-biased. Otherwise, both answers can be incorrect, it is called two-sided errors. Using the previous example (i.e., given an unsorted array). If one wants to review only k elements of the array ($k < N$), in order to reduce the searching time and whether the number is inside of k elements, the algorithm should return *true* (it is correct). On the contrary, if the searched number is in the other portion of elements ($N - k$), the algorithm will respond *false*, when the number is inside the array. This example corresponds to true-biased Monte Carlo algorithms, because when the algorithm returns *true*, it is correct, but if the answer is *false*, it is an incorrect answer.

In our particular case, our algorithm is two-sided errors (it means, that both answers *true* or *false* can be incorrect). It is because, we are not sure about judgments of users with respect to whether a document is either relevant or not relevant regarding the query. Nevertheless, we can suppose that whether the position of document (in the list of documents), is at the top (more close to the query), the document has higher probability to be relevant than a document is at the bottom of the list.

IV. Approach for reusing past queries

This section presents an approach for reusing past queries that fits into the global process introduced in section II. This approach considers a storage process in which each computed query is stored with its documents and their relevance judgments. Second, for the retrieval process, each new query is

checked and compared with the past queries in the system, and if there is a past query quite similar in the system then relevant documents of this past query are recovered according to our algorithm. In this way, relevant documents are used to respond to the new query.

Our algorithm splits the list of retrieved documents in NG groups (see Figure 1). The first group, which is conformed by the first elements appearing at the top of the list, has the greatest probability to get a *hit*. At the same time, each element of this group has 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. On balance, each group has different probabilities to have a *hit*, and at the same time, each element of this group has different likelihoods.

A. Definitions and Notations

Definition 1. Let $DB = \{D \cup Q \mid D = \{d_i\}, Q = \{q_j\}, i, j \in \mathbb{N} \wedge i > 0 \wedge j \geq 0\}$ be an IR dataset, composed of a set of documents D, and a set of past queries Q

a) $\forall d_i \in D \wedge \forall q_j \in Q$, both d_i and q_j are unique.

b) $\bigcap_{j=1}^{|Q|} q_j = \emptyset$

By this definition, we define a collection of documents DB, and a collection of past queries Q. The point b) implies that there are no common terms between past queries.

Definition 2. Let $Q' = \{{q'}_j \mid j \geq 0 \wedge j \in \mathbb{N}\}$ be a set of new queries

$\bigcap_{j=1}^{|Q'|} {q'}_j = \emptyset$.

This definition states that there that there are no common terms between new queries.

Definition 3. Let $V_N(q)$ be a set of N retrieved documents given q.

Definition 4. Let $A(q) = \{d_s \mid d_s$ is a relevant document given the query q, $\forall d_s \in V_N(q)\}$ be the set of all the relevant documents retrieved for the query q.

Definition 5. Let $A'(q) = \{d_l \mid d_l$ is not a relevant document, given the query q,$\forall d_l \in V_N(q)\}$ be the set of all the irrelevant documents retrieved for the query q.

Definition 6. $V_N(q) = A(q) \cup A'(q)$.

$V_N(q)$ corresponds to the set of retrieved documents, which gathers relevant and not relevant documents.

Definition 7. Given a query q, such as $q \in Q$, $p_q = (q, V_N(q))$ is defined as the profile of the query q, which the pair composed of the query and its retrieved documents (see section II).

Definition 8. Let $P = \bigcup p_q$ be the set of all the query profiles stored in the system (see section II).

Definition 9. Given a query q', such as $q' \in Q'$. $R_p(q') = \{d \mid d \in V_N(q) \wedge sim(q', q) = max(sim(q', q_i)), \forall p_{q_i} \in P\}$ corresponds to the set of retrieved documents in the profile of the most similar past query according to a similarity measure (e.g., cosine distance).

Definition 10. $\partial : R_p(q') \rightarrow A(q')$ is a function, which assigns the most relevant documents to a new query q', such as $q' \in Q'$ (see *Definition 7* and *Definition 9*).

Definition 11. $\|x\|$ denotes the integer part of a real number.

Definition 12. $\lceil x \rceil$ denotes the upper integer of x.

Definition 13. Given a binary array $B[N]$, such as B has N elements, and $\frac{a}{b}$ is the proportion of values in B that are equal to 1 (true).

This array is the baseline to provide a level of general probability for all the documents. Nevertheless, the probability of each document according to its position in the list $V_N(q)$, is computed by the algorithms 1 and 2.

Definition 14. Let $M(N) = min\{m \mid m \in \mathbb{N} \wedge \sum_{k=0}^{m} 2^k \geq N \wedge N < 2^{m+1}\}$ be the upper bound set, which involves documents of $V_N(q)$ (in power two).

Definition 15. Let $N \simeq NG * 2^{ne}$ be the approximate number of documents, where NG is the number of groups containing documents of $V_N(q)$, and 2^{ne} corresponds to the number of elements by group.

Definition 16. Let i be the position of a document in $V_N(q)$, such as the first element ($i = 1$) represents the most similar document, then
$G_x(ne, i, NG) = min\{x \mid x \in \mathbb{N} \wedge x \in [1, NG] \wedge i \leq x * 2^{ne}\}$, corresponds to the assigned set for the element i.

Definition 17. Let $\varepsilon(ne, i, NG, N) = 1 - [log_2(2^{M(N)} - \langle i \ \ MOD \ \ (G_x(ne, i, NG) + 1)\rangle) - \|log_2(2^{M(N)} - \langle i \ MOD \ (G_x(ne, i, NG) + 1)\rangle\|]$ be the error assigned for the document at the position i in $V_N(q)$.

Definition 18. Let $K(ne, i, N, NG) = \lceil log_2\langle \frac{1}{\varepsilon(ne,i,NG,N)}\rangle \rceil$ be the number of iteration on $B[N]$, to assign the likelihood for the document at the position i in $V_N(q)$.

Thus, $\beta : F(i) \rightarrow r$, is probability function.

$$F(i) = \begin{cases} 1 & : \ Pr_i(1) = \sum_{l=1}^{K(ne,i,N,NG)} \frac{2^{(M(N)-G_x(ne,i,NG))}}{(2^{M(N)})^l} \\ 0 & : \ Pr_i(0) = 1 - Pr_i(1) \end{cases},$$

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

As shown in Figure 1, it is possible to calculate the probability for the document d_1. We setting the values as follows: $N = 30$, $NG = 4$, $ne = 3$, $i = 1$ and the value of $M(N) = 5$. Thus, the probability for the first element of $V_N(q)$ is computed as: $G(3, 1, 4, 30) = 1$ (the first document in the list, is in the first group), $\varepsilon(3, 1, 4, 30) = 0.005$, (see *Algorithm 1*, lines from 12 to 15). Therefore, $K(ne, i, N, NG) = 5$ (see *Algorithm 1*, lines 16 and 17). As the document d_1 belongs to the first group, which has $\frac{1}{2}$, then the probability for the first document is $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32}$ (see *Algorithm 1*, lines from 18 to 25), therefore $Pr_1(1) = 0.968$.

In a similar way, for elements of second group, the search to find a 1 inside the array B, is limited between the range 1 to $\frac{N}{2}$ or $\frac{N}{2} + 1$ to N (see *Algorithm 2* as well as the lines 15 and 16 of *Algorithm 1*). Finally, the probability for the first element of second group d_{13} is $\frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{256} + \frac{1}{1024}$, $Pr_9(1) = 0.333$.

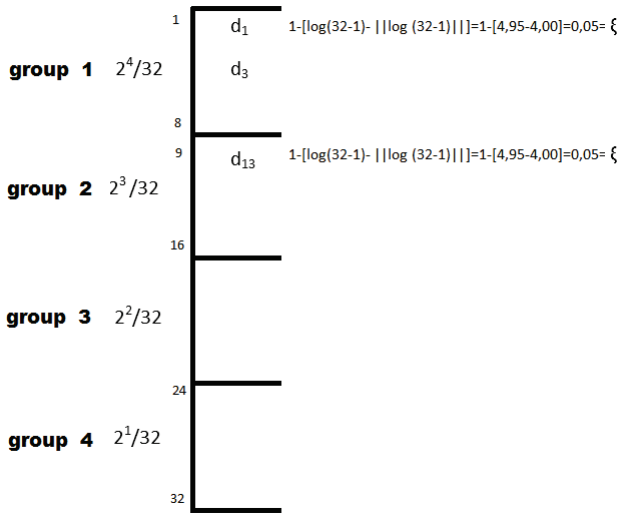

Fig. 1. List of retrieved documents as past search result. Our algorithm splits the list in NG groups ($NG = 4$), with the same quantity of documents ($2^{ne} = 8$). The probability for each document is determined by two factors. First, it depends on the group the document belongs to, and second to the relative position of the document in the group. For example, for the document d_1 (group 1), the error (see *Definition 17*) is 0.05, which is the same for the document d_{13} (group 2). Moreover, the number of iterations K (see *Definition 18*) with the purpose to find 1 inside the $B[N]$ is the same. Nevertheless, the probability in the first group is bounded by $\frac{2^4}{32}$. Finally, the probability for the document in the position i, considering the $group$ (from 1 to NG) , which it belongs, is given by $\sum_{l=1}^{K} \frac{1}{(2^{group})^l}$

Algorithm 1 $B[N], ne, NG, A_{Past}(q), V_N(q), q'$

Require: $B[N]$ is a boolean array, ne is a number of elements in each group, $V_N(q)$ is the list of retrieved documents for the query q, q' is the most similar query with respect to q
Ensure: $A_{New}(q')$ is a list of relevant documents for the query q'

$A_{New}(q) \leftarrow \emptyset$
for $i \leftarrow 1, N$ **do**
  $B[i] \leftarrow false$
end for
for $i \leftarrow 1, \frac{N}{2}$ **do**
  $j \leftarrow random(1, ..., N)$
  $B[j] \leftarrow true$
end for
$i \leftarrow 1$
for $b \leftarrow 1, NG$ **do**
  for $c \leftarrow 1, c < 2^{ne}$ **do**
    $u \leftarrow log_2(2^{M(N)} - c)$
    $U \leftarrow \|u\|$
    $E \leftarrow u - U$
    $E \leftarrow 1 - E$
    $K \leftarrow log_2(\frac{1}{E})$
    $K \leftarrow \ulcorner K \urcorner$
    for $l \leftarrow 1, l \leq K$ **do**
      $index \leftarrow Index(b, N)$
      $indexF \leftarrow (index.Inf + (rand(1, .., index.Sup))$
      if $B[indexF] = true$ **then**
        if $([idDoc = Position(i\ of\ V_N(q))]\ is\ in\ A_{Past}(q))$ **then**
          $A_{New}(q') \leftarrow A_{New}(q') \cup d_{idDoc}$
          $i \leftarrow i + 1$
        end if
      else
        $i \leftarrow i + 1$
      end if
    end for
  end for
end for
$return(A_{New}(q'))$

V. Design of Information Retrieval benchmark

A typical IR collection is composed of three sets: documents, queries and relevance judgments per query (i.e., indications on documents considered as relevant or not relevant) [32]. Our method consists of two steps, based on prior work [33]. The first step aims at creating terms, documents, and queries. Both Heap's law and Zipf's law have been considered to build document collections. Heaps' law indicates that a document with size $O(n)$ (where n is the number of terms) has a vocabulary with size (O^{β}), where $0 < \beta < 1$. Therefore, a simulated document can represent a document written in English of $O(n^2)$ terms [34], [35]. We assume that both processes, elimination of stop words and stemming were carried out. Furthermore, according to the terms that compose each document, documents belong to several subjects. Zipf's law [36], [37] is applied to simulate the distribution of the frequencies of words in the vocabulary (to select the terms from topics). Like Zipf's law, exponential distribution also have been used as an alternative for selecting terms from topics. Otherwise, past queries are created from documents and new queries are built from past queries. Finally, Bradford's law, is used to simulate relevant judgments provided by users about document relevance for a specific query [38].

A. Documents and queries

Each document is composed of terms. A term is made up by letters of the English alphabet. Each letter is come up with uniform distribution and each term is unique. The set of terms is split in subsets that describe topics. The idea is to represent different subjects. Aiming at building a document, either Zipf or Exponential distributions is applied to select terms. According to term selections, each document is related to a main topic and various other minor topics.

Past queries are created from documents. Like documents, terms are chosen under uniform distribution in order to build the past queries. It is important to emphasize that the intersection among past queries is empty. New queries are built from past queries. For each past query a new query is built, either by changing a term or adding another term. Thus, the most similar and unique query for the new query is its corresponding past query. In summary, when a number of queries is given, at the beginning the past queries are built from documents, and afterwards the new queries are created from past queries.

B. Relevance judgments

To represent relevant judgments provided by users about document relevance for a specific query we base on the Bradford's law. The Bradford's law states that most relevant

Algorithm 2 $Index(b, N)$

Require: b is the number of the group, N is the limit
Ensure: $Index$ Upper and Lower bound

$Index.Inf \leftarrow 0$
$Index.Sup \leftarrow 0$
$valor \leftarrow 0$
if $b = 1$ **then**
    $Index.Inf \leftarrow 1$
    $Index.Sup \leftarrow N$
else
    $valor \leftarrow random(0, 1)$
    if $valor = 0$ **then**
        $Index.Inf \leftarrow 1$
        $Index.Sup \leftarrow \frac{N}{2^b}$
    else
        $Index.Inf \leftarrow \frac{N}{2}$
        $Index.Sup \leftarrow \frac{N}{2} + \frac{N}{2^b}$
    end if
end if
$return(Index)$

papers are in few journals while other relevant papers are spread on a high quantity of other journals [38], [39]. By analogy, we assume that in the list of documents returned in response to a submitted query most relevant documents should be at the top of the list while other relevant documents are spread in the remaining of the list.

In our context of reusing past searches, one of our hypotheses concerns similarities between queries and between information needs. Two very similar queries refer to the same information need as usually supposed in information retrieval. Consequently, there is a subset of relevant documents for both queries. However, both queries may have additional distinct relevant documents. For example, in the Figure 2, 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.

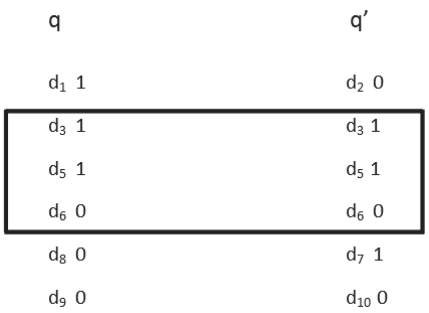

Fig. 2. Relevant documents for each query have 1, irrelevant documents have 0. First, both list of documents are retrieved, from the intersection (d_3, d_5, and d_6), relevant documents common to both queries are computed by using Bradford's law. Afterwards, relevant documents are calculated again for each query by preserving the relevant documents of the intersection (d_3 and d_5).

Our method is based on Zeta distribution which provides a discrete approximation of the Bradford's law. First, Zeta distribution is applied to on a common sublist of documents extracted from he document lists of both queries aiming to determine the common relevant documents for both queries. Second, Zeta distirubtion is applied to each document list of each query, maintaining the relevant documents in common previously determined. Therefore, the two queries have two different lists of relevant documents.

Thus, precision measures, such as P@10 in our case, may be different for the two queries when returning the same 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. A result list may lead to better precision either for the past query or the new one.

VI. Empirical Results

A. Experimental Environment

Experimental environment is setting like follows, the length of a term $\mid t \mid$, was between 3 and 7. Uniform distribution was used to establish the length. The number of terms $\mid T \mid$ is 700 in each experiment. The number of terms for each document can be between 15 and 30. According to Heaps's law [40], it is possible to represent documents between 300 to 900 words in our case. The number of topics used in each experiment is 7. Each topic is formed by 100 terms. When a document is built, terms of other topics are chosen using either Exponential distribution or Zipf distribution. Whereby, most words, which compose a document are chosen form a specific topic. Document numbers used in each experiment were 700, 1400, 2100, 2800 and 3500.

On the other hand, terms for a query were between 3 and 8. Terms of a query are chosen from a particular document. Both terms and documents are chosen using uniform distribution to build the past queries. We have built 15 past queries. From the set of past queries, 15 new queries were built. Finally, the number of queries in each experiment was 30.

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

Simulations were implemented on C language and run on Linux Ubuntu 3.2.9, with Centrino 1350, 1.8 Ghz Intel processor, 1GB RAM, and gcc 4.6.3 compiler.

B. Experimental Results

In this section, three experiments are studied. Both, first and second experiment, Exponential distribution have been used to build the collection of documents D. Like previous, in the third experiment, Zipf distribution has been applied to build D. Additionally, we have used the Student's Paired t-Test (paired samples) over each average P@10 (our approach with respect to traditional retrieval) for each set of documents (700, 1400, 2100, 2800 and 3500). Preliminary results are shown as average P@10 over all the queries.

1) Experiment 1: a) Exponential distribution (with parameter $\theta = 1.0$) was used to build D. Zeta distribution (with parameter $S = 2$) was applied in order to determine the list of relevant documents for each query. Average results for P@10 can be seen in Table I (see $S = 2$). An average of 3.8 queries were not improved applying our approach (with respect to the past query). An average of 25.8 queries led to

TABLE I. EVALUATION RESULTS OF THE TWO APPROACHES (I.E., *Past Results* AND *Cosine*) ACCORDING TO AVERAGE P@10 (OVER 30 QUERIES) WITH A COLLECTION *D* BASED ON EXPONENTIAL DISTRIBUTION USING $\theta = 1.0$

Zeta Distribution	*D* Size	Approach	
		Past Results	Cosine
$S = 2$	700	0.758	0.589
	1400	0.738	0.615
	2100	0.756	0.607
	2800	0.769	0.672
	3200	0.754	0.639
$S = 3$	700	0.642	0.499
	1400	0.649	0.547
	2100	0.608	0.529
	2800	0.652	0.581
	3200	0.611	0.559
$S = 4$	700	0.555	0.429
	1400	0.516	0.429
	2100	0.534	0.428
	2800	0.547	0.486
	3200	0.541	0.468

TABLE II. EVALUATION RESULTS OF THE TWO APPROACHES (I.E., *Past Results* AND *Cosine*) ACCORDING TO AVERAGE P@10 (OVER 30 QUERIES) WITH A COLLECTION *D* BASED ON EXPONENTIAL DISTRIBUTION USING $\theta = 1.5$

Zeta Distribution	*D* Size	Approach	
		Past Results	Cosine
$S = 2$	700	0.764	0.638
	1400	0.787	0.675
	2100	0.716	0.595
	2800	0.784	0.683
	3200	0.736	0.634
$S = 3$	700	0.637	0.556
	1400	0.680	0.597
	2100	0.654	0.511
	2800	0.684	0.590
	3200	0.646	0.558
$S = 4$	700	0.564	0.499
	1400	0.564	0.499
	2100	0.570	0.469
	2800	0.578	0.506
	3200	0.531	0.451

result improvements with our approach. The highest p-value was 4.234 E-25 applying the Paired t-test.

b) Here, Exponential distribution ($\theta = 1.0$) was applied to build D. Zeta distribution ($S = 3$) was used to determine the list of relevant documents for each query. Average results for P@10 are displayed in Table I (see $S = 3$). An average of 7.4 queries were not improved with our approach. An average of 21 queries led to result improvements with our approach. The highest p-value was 1.291 E-14 applying the Paired t-test.

c) In this scenario, Exponential distribution (with parameter $\theta = 1.0$) was used to build D. Zeta distribution (with parameter $S = 4$) was applied to determine the list of relevant documents for each query. In Table I (see $S = 4$), average results for P@10 can be seen. An average of 5.8 queries were not improved with our approach. Our approach improved, on average, 21.1 queries. The highest p-value was 2.245 E-14 applying the Paired t-test.

2) Experiment 2: a) In this scenario, Exponential distribution (with parameter $\theta = 1.5$) was applied to build the collection D. Zeta distribution (with parameter $S = 2$) was used to determine the list of relevant documents for each query. Average results for P@10 are displayed in Table II (see $S = 2$). An average of 4 queries were not improved applying our approach while an average of 25.6 queries led to result improvements with our approach. When calculating the Student's paired t-test, the highest p-value was 5.140 E-20.

b) Exponential distribution (with parameter $\theta = 1.5$) was used to build the collection D. Zeta distribution (with parameter $S = 3$) was used to determine the list of relevant documents for each query. From Table II(see $S = 3$), average results for P@10 can be seen. An average of 5.4 queries were not improved applying our approach. Our approach improved, on average, 24 queries. The highest p-value was 1.063 E-17 for the Paired t-test.

c) In this scenario, Exponential distribution (with parameter $\theta = 1.5$) was applied in order to build the collection D. Zeta distribution (with parameter $S = 4$) was used to determine the list of relevant documents for each query. Average results for P@10 are shown in Table II (see $S = 4$). Our approach did not improve, on average, 8 queries. An average of 19.4 queries were improved with our approach. The highest p-value applying the Paired t-test was 7.104 E-13.

3) Experiment 3: a) In this scenario, Zipf distribution (with parameter $\lambda = 1.6$) was applied to build D. Zeta distribution (with parameter $S = 2$) was used to determine the list of relevant documents for each query. Average results for P@10 are shown in Table III (see $S = 2$). The average number of queries, where our approach did not improve regarding past queries, was 4. The average number of queries where our approach was improved, corresponds to 24.8. The highest p-value was 9.717 E-24 applying the Paired t-test.

b) Here, Zipf distribution (with parameter $\lambda = 1.6$) was used to build D. Zeta distribution (with parameter $S = 3$) was applied to determine the list of relevant documents for each query. As can be seen in Table III (see $S = 3$), average results for P@10 are displayed. The average number of queries where our approach lost with respect to the past query, was 6.8. The average number of queries where our approach was better, corresponds to 22.4. The highest p-value was 1.956 E-17 applying the Paired t-test.

c) Zipf Distribution (with parameter $\lambda = 1.6$) was used to build D. Zeta distribution (with parameter $S = 4$) was applied to determine the list of relevant documents for each query. Average results for P@10 are shown in Table III (see $S = 4$). The average number of queries, where our approach did not improve, corresponds to 3.6. The average number of queries where our approach was better, was 25.2. The highest p-value was 2.316 E-23 applying the Paired t-test.

TABLE III. EVALUATION RESULTS OF THE TWO APPROACHES (I.E., *Past Results* AND *Cosine*) ACCORDING TO AVERAGE P@10 (OVER 30 QUERIES) WITH A COLLECTION *D* BASED ON ZIPF DISTRIBUTION USING $\lambda = 1.6$

Zeta Distribution	*D* Size	Approach	
		Past Results	Cosine
$S = 2$	700	0.748	0.660
	1400	0.758	0.607
	2100	0.736	0.622
	2800	0.767	0.599
	3200	0.727	0.604
$S = 3$	700	0.599	0.523
	1400	0.659	0.570
	2100	0.625	0.550
	2800	0.673	0.531
	3200	0.650	0.549
$S = 4$	700	0.526	0.450
	1400	0.547	0.454
	2100	0.543	0.454
	2800	0.573	0.449
	3200	0.561	0.434

C. Discussion

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 the case of Exponential distribution, the distribution with value 0.5 was omitted because we considered that it would not provide relevant results, after of obtaining the results whene using the values 1.5 and 1.0.

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

It should be noted that if the number of queries was increased in these experiments, final results should not be different. This is because for every past query, it exists only one and unique query for which the intersection is not empty.

In addition, it should be noticed that our algorithm provided the best average P@10 in every experiment, with p-values always supporting statistically significant results.

VII. CONCLUSION AND FUTURE WORK

We proposed in this paper an approach for reusing past queries to respond to a new submitted query. The approach relies on two processes: a storage process and a retrieval process. This retrieval is based on a probabilistic approach. A new Monte Carlo algorithm have been proposed to select relevant documents for a new query from its most similar past query. This algorithm is easy to implement, does not require time of learning, and provides acceptable results improving precision. Furthermore, this algorithm can be implemented not only inside some IRS but also as external interface outside search engines. This algorithm aims at reusing relevant documents retrieved from the most similar past query. In addition, different evaluation scenarios have been simulated. Simulation provided an appropriate environment to evaluate our algorithm. Documents, queries, and relevance judgments on documents given a query were simulated. Different experiments have been carried out on several probability distributions to cover various types of collections. Empirical results showed better precision (P@10) of our algorithm compared to a traditional retrieval approach (i.e., cosine).

Future work will be devoted to define more real evaluations relying on past queries by adapting TREC collections for instance. We will also develop other approaches to construct retrieved documents for a new query by combining various results of past queries.

REFERENCES

[1] M. J. Kearns, "The computational complexity of machine learning," Ph.D. dissertation, Cambridge, MA, USA, 1989, UMI Order No: GAX89-26128.

[2] Y. Jin and B. Sendhoff, "Reducing fitness evaluations using clustering techniques and neural network ensembles," in *Genetic and Evolutionary Computation GECCO 2004*, ser. Lecture Notes in Computer Science, K. Deb, Ed. Springer Berlin Heidelberg, 2004, vol. 3102, pp. 688–699.

[3] Z. M. Nopiah, M. I. Khairir, S. Abdullah, M. N. Baharin, and A. Arifin, "Time complexity analysis of the genetic algorithm clustering method," in *Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation*, ser. ISPRA'10. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 171–176.

[4] M. Sanderson, "Test Collection Based Evaluation of Information Retrieval Systems," *Foundations and Trends in Information Retrieval*, vol. 4, no. 4, pp. 247–375, Aug. 2010.

[5] O. Alonso and S. Mizzaro, "Using crowdsourcing for trec relevance assessment," *Inf. Process. Manage.*, vol. 48, no. 6, pp. 1053–1066, Nov. 2012.

[6] S. Cetintas, L. Si, and H. Yuan, "Using past queries for resource selection in distributed information retrieval," Department of Computer Science, Purdue University, Tech. Rep. 1743, 2011. [Online]. Available: http://docs.lib.purdue.edu/cstech/1743

[7] R. Cen, Y. Liu, M. Zhang, B. Zhou, L. Ru, and S. Ma, "Exploring relevance for clicks," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, ser. CIKM '09. New York, NY, USA: ACM, 2009, pp. 1847–1850.

[8] L. Azzopardi, K. Järvelin, J. Kamps, and M. D. Smucker, "Report on the sigir 2010 workshop on the simulation of interaction," *SIGIR Forum*, vol. 44, no. 2, pp. 35–47, Jan. 2011.

[9] S. Klink, "Improving document transformation techniques with collaborative learned term-based concepts," in *Reading and Learning*, ser. Lecture Notes in Computer Science, A. Dengel, M. Junker, and A. Weisbecker, Eds. Springer Berlin Heidelberg, 2004, vol. 2956, pp. 281–305.

[10] M. Potey, D. Patel, and P. Sinha, "A survey of query log processing techniques and evaluation of web query intent identification," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, Feb 2013, pp. 1330–1335.

[11] X. Shen and C. X. Zhai, "Exploiting query history for document ranking in interactive information retrieval," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ser. SIGIR '03. New York, NY, USA: ACM, 2003, pp. 377–378.

[12] X. Shen, B. Tan, and C. Zhai, "Context-sensitive information retrieval using implicit feedback," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '05. New York, NY, USA: ACM, 2005, pp. 43–50.

[13] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani, "Using association rules to discover search engines related queries," in *Proceedings of the First Conference on Latin American Web Congress*, ser. LA-WEB '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 66–.

[14] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," in *Proceedings of the 2004 International Conference on Current Trends in Database Technology*, ser. EDBT'04. Berlin, Heidelberg: Springer-Verlag, 2004, pp. 588–596.

[15] B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell, "Exploiting query repetition and regularity in an adaptive community-based web search engine," *User Modeling and User-Adapted Interaction*, vol. 14, no. 5, pp. 383–423, Jan. 2005.

[16] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts, "Information re-retrieval: repeat queries in yahoo's logs," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 151–158.

[17] S. Garcia, "Search engine optimisation using past queries," Ph.D. dissertation, RMIT University, Australia, 2007.

[18] P. Clough and M. Sanderson, "Evaluating the performance of information retrieval systems using test collections," *Information Research*, vol. 18, no. 2, 2013. [Online]. Available: http://InformationR.net/ir/18-2/paper582.html

[19] B. Huurnink, K. Hofmann, M. De Rijke, and M. Bron, "Validating query simulators: an experiment using commercial searches and purchases," in *Proceedings of the 2010 international conference on Multilingual and multimodal information access evaluation: cross-language evaluation forum*, ser. CLEF'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 40–51.

[20] J. M. Tague and M. J. Nelson, "Simulation of user judgments in bibliographic retrieval systems," in *Proceedings of the 4th annual international ACM SIGIR conference on Information storage and retrieval: theoretical issues in information retrieval*, ser. SIGIR '81. New York, NY, USA: ACM, 1981, pp. 66–71.

[21] L. Azzopardi, M. de Rijke, and K. Balog, "Building simulated queries for known-item topics: an analysis using six european languages," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 455–462.

[22] V. Dang and B. W. Croft, "Query reformulation using anchor text," in *Proceedings of the third ACM international conference on Web search and data mining*, ser. WSDM '10. New York, NY, USA: ACM, 2010, pp. 41–50.

[23] R. Baeza-Yates, C. Castillo, M. Marin, and A. Rodriguez, "Crawling a country: better strategies than breadth-first for web page ordering," in *Special interest tracks and posters of the 14th international conference on World Wide Web*, ser. WWW '05. New York, NY, USA: ACM, 2005, pp. 864–872.

[24] M. Marin, V. Gil-Costa, C. Bonacic, R. Baeza-Yates, and I. D. Scherson, "Sync/async parallel search for the efficient design and construction of web search engines," *Parallel Comput.*, vol. 36, no. 4, pp. 153–168, Apr. 2010.

[25] A. A. A. Radwan, B. A. A. Latef, A. M. A. Ali, and O. A. Sadek, "Using genetic algorithm to improve information retrieval systems," *World Academy of Science, Engineering and Technology*, vol. 17, pp. 1021–1027, 2008.

[26] G. Salton and C. Buckley, "Readings in information retrieval," K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ch. Improving retrieval performance by relevance feedback, pp. 355–364.

[27] D. Lillis, F. Toolan, A. Mur, L. Peng, R. Collier, and J. Dunnion, "Probability-based fusion of information retrieval result sets," *Artif. Intell. Rev.*, vol. 25, no. 1-2, pp. 179–191, Apr. 2006.

[28] J. A. Shaw and E. A. Fox, "Combination of multiple searches," in *The Second Text REtrieval Conference (TREC-2*, 1994, pp. 243–252.

[29] A. Trotman, "An artificial intelligence approach to information re-trieval," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 603–603.

[30] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 143–151.

[31] E. P. Chan, S. Garcia, and S. Roukos, "Probabilistic modeling for information retrieval with unsupervised training data." in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press, 1998, pp. 159–163.

[32] E. M. Voorhees and D. K. Harman, *TREC: Experiment and Evaluation in Information Retrieval*. Cambridge, MA, USA: MIT Press, 2005.

[33] C. Gutiérrez-Soto and G. Hubert, "Evaluating the interest of revamping past search results," in *Database and Expert Systems Applications*, ser. Lecture Notes in Computer Science, H. Decker, L. Lhotsk, S. Link, J. Basl, and A. Tjoa, Eds. Springer Berlin Heidelberg, 2013, vol. 8056, pp. 73–80.

[34] G. Navarro, E. S. De Moura, M. Neubert, N. Ziviani, and R. Baeza-Yates, "Adding compression to block addressing inverted indexes," *Inf. Retr.*, vol. 3, no. 1, pp. 49–77, Jul. 2000.

[35] E. Silva de Moura, G. Navarro, N. Ziviani, and R. Baeza-Yates, "Fast and flexible word searching on compressed text," *ACM Trans. Inf. Syst.*, vol. 18, no. 2, pp. 113–139, Apr. 2000.

[36] V. Poosala, "Zipf's law," Bell Laboratories, Tech. Rep. 900 839 0750, 1997.

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

[38] E. Garfield, "Bradford's Law and Related Statistical Patterns," *Essays of an Information Scientist*, vol. 4, no. 19, pp. 476–483, May 1980. [Online]. Available: http://www.garfield.library.upenn.edu/essays/v4p476y1979-80.pdf

[39] I. L. Zerchaninova, "Bradford's law of scattering for climate-friendly technologies and metainformational effect of time," Institute for Time Nature Explorations, Tech. Rep., 2008. [Online]. Available: http://www.chronos.msu.ru/EREPORTS/zerchaninova_law.pdf

[40] H. S. Heaps, *Information Retrieval: Computational and Theoretical Aspects*. Orlando, FL, USA: Academic Press, Inc., 1978.