Probabilistic Reuse of Past Search Results

Claudio Gutiérrez-Soto[1,2] and Gilles Hubert[1]

[1] Université de Toulouse, IRIT UMR 5505 CNRS
118 route de Narbonne, F-31062 Toulouse cedex 9
[2] Departamento de Sistemas de Información
Universidad del Bío-Bío, Chile

Abstract. In this paper, a new Monte Carlo algorithm to improve precision of information retrieval by using past search results is presented. Experiments were carried out to compare the proposed algorithm with traditional retrieval on a simulated dataset. In this dataset, documents, queries, and judgments of users were simulated. Exponential and Zipf distributions were used to build document collections. Uniform distribution was applied to build the queries. Zeta distribution was utilized to simulate the Bradford's law representing the judgments of users. Empirical results show a better performance of our algorithm compared with traditional retrieval.

1 Introduction

A wide range of approximations in information retrieval (IR) are devoted to improving the list of documents retrieved to answer particular queries. Among these approaches, we can find solutions that involve efficient assignments of systems to respond to certain types of queries, by applying data mining techniques [1]. Nonetheless, some tasks of data mining can imply not only long periods of time, but also a high cost in money [2]. In addition, solutions that involve an exhaustive analysis of all possible alternatives to find the best answer to a query (i.e., the best precision for each type of query) can be found in IR context. Prior solutions correspond to approaches based on learning techniques (e.g., neural networks, genetic algorithms, and machines support vectors). However, these approaches should imply a high cost in learning time as well as diverse convergence times when the datasets used are heterogeneous [3]. Additionally, characteristics, such as the scopes where these types of algorithms are applied and the performance achieved in different environments, are complex to address [4].

In the IR literature, two types of approaches used in the context of past queries are easily identifiable. The first approaches are based on TREC collections. Most of these approaches use simulation to build similar queries with the aim to provide a suitable framework of evaluation. The second type of approaches rooted in the use of historical queries on the Web, most of which are supported on repetitive queries. As a result, having ad-hoc collections which allow to evaluate the use of past queries in an appropriate way, is a hard task. Therefore, one way to provide an ad-hoc environment for approximations based on past queries is simulation.

Our main contribution is a Monte Carlo algorithm, which uses relevant documents from the most similar past query to answer a new query. The algorithm splits the list of retrieved documents from the most similar past query in subsets of documents. Our algorithm is simple to implement and effective. Moreover, it does not require learning time. Documents, query collections, and relevance judgments of users were simulated to built a dataset for evaluating the performance of our algorithm. A wide range of experiments have been carried out. We have applied the Student's paired t-test to support the experimental results. Empirical results show better results of our algorithm (in particular the precision P@10) than traditional retrieval.

The paper is organized as follows. In section 2, related works on past searches, randomized algorithms, and simulation in IR context are presented. In section 3, we present our approach to simulate an IR collection, in the context of past search results. Section 4 details our approach using past search results, with mathematical definitions. In section 5, empirical results are described. Finally, conclusions are presented in section 6.

2 Related Work

Two categories of approaches employed in the context of past queries are easily identifiable. The first category is based on TREC collections. In a recent work [5], a distributed approach is presented in the context of past queries. Similar queries are simulated from a traditional set of queries. Moreover, the judgments of users are omitted. In [6], two strategies aiming at improving precision were implemented. The first strategy corresponds to the combination of results from previous queries, meanwhile the second implies the combination of query models. An extended work is exposed in [7]. The authors address models based on implicit feedback information to increase precision. Implicit feedback information is given by queries and clickthrough history in an active session. It is important to emphasize that TREC collections used here have been modified to evaluate approximations based on past queries.

The second category of approaches focuses on log files in the context of the Web. In [8], an automatic method to produce suggestions based on previously submitted queries is presented. To achieve this goal, an algorithm of association rules was applied on log files. The 95 most popular queries were considered. Nonetheless, the percentage of these 95 queries over 2.3 millions of records is unknown. Hence, it is infeasible to estimate the impact of this approximation. Moreover, [9] claims that there is no easy way to calculate the real effect of approximations founded on association rules. It is mainly due to the complexity to determine the successive queries that belong to the same session (i.e., for the same user). In [10], an approximation based on repeated queries is exposed. The aim is the identification of identical queries executed in the same trace. In [11], two contributions, which take advantages from repeated queries, are presented. The first contribution is aligned on efficiency in execution time and the second is focused on repetitive document access by the search engines.

Simulation to evaluate information retrieval systems (IRSs) is presented as a novel branch of research [12]. Simulation in IR is an automatic method, where documents, query collections, and judgments of users can be built without user intervention [13].

In addition, the IR literature is crammed with contributions based on probabilistic algorithms. The major part of probabilistic algorithms in IR can be categorized in two classes, learning techniques and optimization. Typically, approximations rooted in learning techniques involve the use of Bayesian Networks and their variants. The PrTFIDF algorithm, which is a probabilistic version of TFIDF algorithm is presented in [14]. PrTFIDF provides a new vision of vector space model, where the theorem of total probability, the Bayes' theorem, and a descriptor for every document are used. Final results show a better performance than TFIDF. In [15], the classification of documents in an unsupervised manner is carried out. It uses Poisson distribution according to the query or topic.

Several optimization techniques involve the use of Genetic Algorithms (GA). Inspired by the formula proposed by Salton [16] (where the term weights for documents and queries are the product between the term frequency multiplied by an inverse collection frequency factor), a new fitness function is presented in [17]. Both, vectors of documents and queries are normalized by using the formula. Experimental results show better effectiveness when using this approach than traditional retrieval (i.e., using cosine distance). Eventually, the *Probfuse* algorithm proposed in [18], whose aim is to combine results from several IR algorithms, outperforms the widely used CombMNZ algorithm.

Different to Bayesian Networks and GA, Monte Carlo and Las Vegas algorithms are used usually when the problem is hard to solve like NP problems or when algorithm input is non-deterministic. Las Vegas algorithms provide an answer, which is always correct and where in the worst case the execution time is the same as the deterministic version. In contrast to Las Vegas algorithms, Monte Carlo algorithms give an answer, which can be incorrect (i.e., the algorithm returns *true*, when the answer should be *false*, or vice-versa). When one of these answers is correct, it is called true-biased (the correct answer is *true*) or false-biased (the correct answer is *false*). When both answers can be incorrect, it is called two-sided errors.

Our Monte Carlo algorithm corresponds to the type *two-sided errors.* This is due to the fact that we are not sure about judgments of users with respect to whether a document is either relevant or not relevant regarding the query. Nonetheless, we assume that documents that appear at the top of the result list have more probability to be relevant than documents that appear at the bottom of the list.

3 Simulating IR Collections

Our method consists of two steps, based on prior work [19]. The first step aims at creating terms, documents, and queries. Both Heaps' and Zipf's laws are considered to build document collections. We assume that both processes, elimination of stop words, and stemming were carried out. Due to terms which compose a document can belong to several subjects, Zipf's law is applied to select terms from topics [20]. Exponential distribution can be applied as an alternative to Zipf's law. Then, past queries are created from documents and new queries are built from past queries. In the final step, to simulate judgments provided by users about relevance of documents for a specific query, Bradford's law is applied [21].

The most basic element that composes a document is a term. A term is composed of letters from the English alphabet. Both documents and queries are composed of terms. Each document is unique. Past queries are built from documents and their intersections are empty. A topic (i.e., subject of documents) is defined by terms. Several topics are used to built a document. The intersection among topics is empty. Aiming to build documents, Zipf and Exponential distributions are used to select terms from different topics. Uniform distribution is used to select documents, where terms are selected to built the past queries. Then, past queries are built from the documents. New queries are made up from past queries by either adding a new term or deleting a term. Bradford's law has been applied through Zeta distribution. In order to explain the mechanism to obtain the precision, we can assume two lists of documents. The first list of documents (for the query 1) is composed of the documents $d_1(1), d_3(1), d_5(1), d_6(0), d_8(0), d_9(0)$ where $d_i(1)$ is a relevant document, meanwhile $d_i(0)$ is irrelevant. In the same way, the second list of documents is composed by the documents $d_2(0), d_3(1), d_5(1), d_6(0), d_7(1), d_{10}(0)$. Thus, first, a subset of common documents is found ($d_3,\ d_5$ and d_6). Second, from the common subset, relevant documents are determined for both queries by using Bradford's law (d_3 and d_5). Third, Bradford's law is applied for each list by conserving the relevant documents that belong to the common subset (for the first list $d_1,\ d_3$ and d_5 are relevant documents). As a consequence, the precision (in our case P@10) is different for both queries.

4 Retrieval Using Past Queries

At the beginning, each submitted query is saved with its documents. Afterwards, each new query is compared with the past queries stored in the system. If there is a past query quite similar, then the relevant documents are retrieved from the most similar past query using our algorithm. Broadly speaking, our algorithm divides the list of documents retrieved from the past query, in groups of power two. For example, if the list of documents comprises 30 documents, the number of documents will be rounded up to the next number in power two, i.e., $n = 32$. Later on, groups of documents are defined as follows. The first group comprises 2^0 documents. The second group involves 2^1 documents, the third group is composed of 2^2 documents, and so on, in such a way that the sum of the documents does not outperform $n = 32$. Thus, the number of groups is 5. The biggest group is composed of documents that appear in the first positions (between the position 1 and 16). The next biggest group comprises documents that appear from the position 17 to 24, and so on. The likelihood of a document to be relevant is determined by two factors: the group it belongs to and its position in the group. The algorithm and a more detailed example are displayed in the next sections.

4.1 Definitions and Notations

Let DB be an IR dataset, composed of a set of documents D, and a set of past queries Q. Besides, let Q' be the set of new queries. $V_N(q)$ is a set of N retrieved documents given q, and $sim(q, d_j)$ is the cosine distance between query q and the document d_j. Besides, $V_N(q) = A(q) \cup A'(q)$, where $A(q)$ corresponds to the set of all relevant documents for the query q. $A'(q)$ is the set all irrelevant documents for the query q. $C = \bigcup c(q, V_N(q))$ is the set of all retrieved documents with their respective queries.

Definition 1. *$\partial : R_c(q') \to A(q')$ is a function, which assigns the most relevant documents to the new query q', such as $q' \in Q'$ and $R_c(q')$ corresponds to a set of retrieved documents, from the most similar past query. (see Definition 2 and Definition 4).*

In addition, let $\|x\|$ be the integer part of a real number x, $\lceil x \rceil$ corresponds to the upper integer of x and $\lfloor x \rfloor$ corresponds to the lower integer of x. $B[N]$ is a binary array such as B has N elements, and $\frac{a}{b}$ is the proportion of values in B (see *Algorithms 1*, lines from 9 to 12), which have the value 1 (true). This array is the base to provide a level of general probability for all documents. Nevertheless, the probability of each document according to the position in $V_N(q)$, is computed by the *Algorithm 1*.

Definition 2. *$M(N) = min\{m \mid m \in \mathbb{N} \wedge \sum_{k=0}^{m} 2^k \geq N \wedge N < 2^{m+1}\}$ be the upper bound set, which involves documents of $V_N(q)$(in power two) (see Algorithm 1, lines from 2 to 4).*

Definition 3. *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 $f_x(i, N) = min\{x \mid x \in \mathbb{N} \wedge i \leq \sum_{k=1}^{x} \frac{2^{M(N)}}{2^k}\}$, corresponds to the number of set assigned for the document i (see Algorithm 1, line 5).*

Definition 4. *Let $v(i, N) = (2^{M(N)-f_x(i,N)} - 1) - [\langle(\sum_{k=1}^{f_x(i,N)} 2^{M(N)-k}) - i\rangle mod(2^{M(N)-f_x(i,N)})]$ be the value assigned to i, from 0 to $2^{M(N)-f_x(i,N)}$ (see Algorithm 1, line 15).*

Definition 5. *$\Phi(i, N) = log_2(2^{M(N)-f_x(i,N)} - v(i, N)) - \|log_2(2^{M(N)-f_x(i,N)} - v(i, N))\|$ a decimal number, which is $[0, 1[$ (see Algorithm 1, lines from 16 to 19).*

Definition 6. *Let* $K(i,N) = \begin{cases} \lceil \Phi(i,N) \rceil * \|M(N) - f_x(i,N)\| & : \; if \quad \Phi(i,N) \geq 0.5 \\ \lfloor \Phi(i,N) \rfloor * \|M(N) - f_x(i,N)\| & : \; if \quad \Phi(i,N) < 0.5 \end{cases}$ *be the number of iterations to look for a hit in the array B (see Algorithm 1, lines from* 20 *to* 25*).*

Thus, $\beta : F(i) \rightarrow \{0, 1\}$, is the *hit* and *miss* function.

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

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

Our algorithm works as follows. The list of retrieved documents is split in subsets of elements in power two. In our case, $V_N(q)$ has 30 documents, however it can be approximated to 32 documents. Thus, if we apply *Definition 2*, then $M(N) = 5$. Therefore, $V_N(q)$ is split in 5 subsets. In general terms, $Pr_i(1)$ for every subset is different. Specifically on $\frac{2^{\|M(N)-f_x(i,N)\|}}{(2^{M(N)})}$. Thus, the space of possible candidates for the first subset is $\frac{2^{\|5-1)\|}}{(2^5)} = \frac{1}{2}$, for the second subset is $\frac{2^{\|5-2)\|}}{(2^5)} = \frac{1}{4}$ and so on. To show how the probability decreases according to the subsets, two examples are provided, for the first subset and the third subset. The second element of the first subset is $i = 2$, thus applying the *Definition 3*, $f_x(2, 30) = 1$, therefore, $v(2, 30) = (2^{5-1} - 1) - [\langle(\sum_{k=1}^{1} 2^{5-1}) - 2\rangle) \; mod \; 2^{5-1}] = 1$ (see *Definition 4*). Applying *Definition 5*.

$\Phi(2, 30) = log_2(2^{5-1} - 1) - \|log_2(2^{5-1} - 1)\| = 3.906 - 3 = 0.906$. Applying *Definition 11*, $K(2, 30) = \lceil \Phi(2, 30) \rceil * \|5 - 1\| = 1 * 4 = 4$.

Thus, $Pr_2(1) = \sum_{l=1}^{4} \frac{2^{\|5-1\|}}{(2^5)^l} = 0.757$. In the same way, $v(26, 30) = (2^{5-3} - 1) - [\langle(16 + 8 + 4) - 26\rangle) \; mod \; 2^{5-3}] = 1$. Finally, $Pr_{26}(1) = \sum_{l=1}^{2} \frac{2^{\|5-3\|}}{(2^5)^l} = 0.128$

5 Experiments

5.1 Experimental Environment and Empirical Results

The experimental environment was instantiated as follows. The length of a term is between 3 and 7. The length was determined using Uniform distribution. The total number of terms used in each experiment corresponds to 800. A document can contain between 15 and 30 terms. The number of topics used in each experiment is 8. Each topic is defined by 100 terms. Each experiment used 800, 1600, 2400, 3200, and 4000 documents. Terms for a query were between 3 and 8. We built 15 past queries from documents. From the set of past queries, 15 new queries were built. Thereby, we used 30 distinct 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.

Three experimental scenarios were defined. For the first and second experiments, Exponential distributions (with parameters $\theta = 1.0$ and $\theta = 1.5$) were

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

```
Require: $B[N]$ is a boolean array, $A_{Past}(q)$ is a set of relevant documents for the
    query $q$, $V_N(q)$ is the set of retrieved documents for the query $q$, $q'$ is the most
    similar query for $q$
Ensure: $A_{New}(q')$ is a set of relevant documents for the query $q'$
 1: $A_{New}(q) \leftarrow \emptyset$
 2: for $i \leftarrow 0, sum \leftarrow 0, sum < N+1$ do
 3:     $sum \leftarrow sum + 2^i$
 4: end for
 5: $k \leftarrow i - 1$
 6: for $i \leftarrow 1, N$ do
 7:     $B[i] \leftarrow false$
 8: end for
 9: for $i \leftarrow 1, \frac{N}{2}$ do
10:     $j \leftarrow random(1, ..., N)$
11:     $B[j] \leftarrow true$
12: end for
13: $l \leftarrow 1$
14: while do $k \geq 0$ $AND$ $l < N$
15:     for $i \leftarrow 0, i < 2^k$ do
16:         $I \leftarrow 2^k - i$
17:         $u \leftarrow log_2(I)$
18:         $U \leftarrow \|u\|$
19:         $u \leftarrow u - U$
20:         if $u - 0,5 \geq 0$ then
21:             $K \leftarrow \lceil log_2(I) \rceil$
22:         else
23:             $K \leftarrow \lfloor log_2(I) \rfloor$
24:         end if
25:         for $j \leftarrow 1, j \leq k * K$ do
26:             if $2^k * 2 \geq N$ then
27:                 $index = N$
28:             else
29:                 $index = 2^k * 2 - 1$
30:             end if
31:             if $B[index] = true$ then
32:                 if $([idDoc = Position(l\ of\ V_N(q))]\ is\ in\ A_{Past}(q))$ then
33:                     $A_{New}(q') \leftarrow A_{New}(q') \cup d_{idDoc}$
34:                     $l \leftarrow l + 1$
35:                 end if
36:             else
37:                 $l \leftarrow l + 1$
38:             end if
39:         end for
40:     end for
41:     $k \leftarrow k - 1$
42: end while
43: $return(A_{New}(q'))$
```

applied to build the collection of documents D. In the third experiment, Zipf distribution (with parameter $\theta = 1.6$) was applied to build D. Simulations of user judgments were carried out under Zeta Distribution. Zeta distribution with parameters $2, 3$, and 4 were applied on the 30 most similar documents with respect to the queries. Besides, the Student's Paired t-Test (Two Samples test) over each average P@10 (our approach with respect to traditional retrieval) were used to support the results. Final results are summarized and displayed in Table 1.

Table 1. Results comparing our approach of reusing past queries with cosine distance

Experiment Distribution to build collection D	Relevance simulation Zeta distribution with parameter S	Percentage of improved queries (Measure: P@10)	Average improvement (Measure: P@10)
Experiment 1	S = 2	83 %	+21 %**
Exponential distribution	S = 3	74 %	+17 %**
(with parameter $\theta = 1.0$)	S = 4	65 %	+20 %**
Experiment 2	S = 2	77 %	+17 %**
Exponential distribution	S = 3	73 %	+18 %**
(with parameter $\theta = 1.5$)	S = 4	65 %	+16 %**
Experiment 3	S = 2	85 %	+21 %**
Zipf distribution	S = 3	84 %	+18 %**
(with parameter $\theta = 1.6$)	S = 4	77 %	+23 %**

** p-value<0.01 (two sample t-test).

5.2 Discussion

Accepted ranges for Zipf's law regarding the distribution of word frequencies in a vocabulary are between 1.4 and 1.8. In our experiments, Zipf distribution was used with value 1.6 to select terms from topics. It is important to emphasize that every time the parameter S of Zeta distribution (to apply Bradford's law) was incremented, both averages of P@10, using *Past Result* (i.e., our approach) and *Cosine* (i.e., traditional IR) declined similarly. Additionally, if the number of queries was increased in the experiments, it should not have different final results. This is because for every past query it exists just one and unique query for which the intersection is not empty. Also, the final results show that increasing the number of documents have no impact on the significance test p-values.

6 Conclusions

In this paper, a new Monte Carlo algorithm for information retrieval using past queries have been presented. It is easy to implement, does not require time to learning, and provides acceptable results improving precision (i.e., P@10). Furthermore, this algorithm can be implemented not only inside an information retrieval system but also as external interface outside search engines. This algorithm relies on reuse of relevant documents retrieved from the most similar past query. In addition, different evaluation scenarios have been simulated. Simulation provides two advantages. First, it provides an ideal environment to evaluate our algorithm. Second it makes possible to build not only document and query collections but also relevance judgments for documents given a query. Empirical results showed better precision (P@10) of our algorithm compared with traditional retrieval.

References

1. Bigot, A., Chrisment, C., Dkaki, T., Hubert, G., Mothe, J.: Fusing different information retrieval systems according to query-topics: a study based on correlation in information retrieval systems and trec topics. Inf. Retr. 14(6), 617–648 (2011)
2. Gray, P., Watson, H.J.: Present and future directions in data warehousing. SIGMIS Database 29(3), 83–90 (1998)
3. Nopiah, Z.M., Khairir, M.I., Abdullah, S., Baharin, M.N., Arifin, A.: Time complexity analysis of the genetic algorithm clustering method. In: Proceedings of the 9th WSEAS International Conference on Signal Processing, Robotics and Automation, ISPRA 2010, Stevens Point, Wisconsin, USA, pp. 171–176. World Scientific and Engineering Academy and Society, WSEAS (2010)
4. Kearns, M.J.: The Computational Complexity of Machine Learning. PhD thesis, Harvard University, USA, Cambridge, MA, USA (1989)
5. Cetintas, S., Si, L., Yuan, H.: Using past queries for resource selection in distributed information retrieval. Technical Report 1743, Department of Computer Science, Purdue University (2011)
6. Shen, X., Zhai, C.X.: 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, SIGIR 2003, pp. 377–378. ACM, New York (2003)
7. Shen, X., Tan, B., Zhai, C.: Context-sensitive information retrieval using implicit feedback. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2005, pp. 43–50. ACM, New York (2005)
8. Fonseca, B.M., Golgher, P.B., de Moura, E.S., Ziviani, N.: Using association rules to discover search engines related queries. In: Proceedings of the First Conference on Latin American Web Congress, LA-WEB 2003, pp. 66–71. IEEE Computer Society, Washington, DC (2003)
9. Baeza-Yates, R., Hurtado, C., Mendoza, M.: Query recommendation using query logs in search engines. In: Lindner, W., Fischer, F., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 588–596. Springer, Heidelberg (2004)
10. Teevan, J., Adar, E., Jones, R., Potts, M.A.S.: 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, SIGIR 2007, pp. 151–158. ACM, New York (2007)
11. Garcia, S.: Search Engine Optimisation Using Past Queries. PhD thesis, RMIT University, Australia (2007)
12. Clough, P., Sanderson, M.: Evaluating the performance of information retrieval systems using test collections. Information Research 18(2) (2013)

13. Huurnink, B., Hofmann, K., de Rijke, M., Bron, M.: Validating query simulators: An experiment using commercial searches and purchases. In: Agosti, M., Ferro, N., Peters, C., de Rijke, M., Smeaton, A. (eds.) CLEF 2010. LNCS, vol. 6360, pp. 40–51. Springer, Heidelberg (2010)
14. Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning, ICML 1997, pp. 143–151. Morgan Kaufmann Publishers Inc., San Francisco (1997)
15. Chan, E.P., Garcia, S., Roukos, S.: Probabilistic modeling for information retrieval with unsupervised training data. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD), pp. 159–163. AAAI Press (1998)
16. Salton, G., Buckley, C.: Readings in information retrieval. In: Sparck Jones, K., Willett, P. (eds.) Readings in Information Retrieval, pp. 355–364. Morgan Kaufmann Publishers Inc., San Francisco (1997)
17. Radwan, A.A.A., Latef, B.A.A., Ali, A.M.A., Sadek, O.A.: Using genetic algorithm to improve information retrieval systems. World Academy of Science, Engineering and Technology 17, 1021–1027 (2008)
18. Lillis, D., Toolan, F., Mur, A., Peng, L., Collier, R., Dunnion, J.: Probability-based fusion of information retrieval result sets. Artif. Intell. Rev. 25(1-2), 179–191 (2006)
19. Gutiérrez-Soto, C., Hubert, G.: Evaluating the interest of revamping past search results. In: Decker, H., Lhotská, L., Link, S., Basl, J., Tjoa, A.M. (eds.) DEXA 2013, Part II. LNCS, vol. 8056, pp. 73–80. Springer, Heidelberg (2013)
20. Poosala, V.: Zipf's law. Technical Report 900 839 0750, Bell Laboratories (1997)
21. Garfield, E.: Bradford's Law and Related Statistical Patterns. Essays of an Information Scientist 4(19), 476–483 (1980)