Generating suggestions for queries in the long tail with an inverted index

Daniele Broccolo [a,b], Lorenzo Marcon [a], Franco Maria Nardini [a,*], Raffaele Perego [a], Fabrizio Silvestri [a]

[a] *Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR, Pisa, Italy*
[b] *Università "Ca' Foscari" Venezia, Italy*

ABSTRACT

This paper proposes an efficient and effective solution to the problem of choosing the queries to suggest to web search engine users in order to help them in rapidly satisfying their information needs. By exploiting a weak function for assessing the similarity between the current query and the knowledge base built from historical users' sessions, we re-conduct the suggestion generation phase to the processing of a full-text query over an inverted index. The resulting query recommendation technique is very efficient and scalable, and is less affected by the data-sparsity problem than most state-of-the-art proposals. Thus, it is particularly effective in generating suggestions for rare queries occurring in the long tail of the query popularity distribution. The quality of suggestions generated is assessed by evaluating the effectiveness in forecasting the users' behavior recorded in historical query logs, and on the basis of the results of a reproducible user study conducted on publicly-available, human-assessed data. The experimental evaluation conducted shows that our proposal remarkably outperforms two other state-of-the-art solutions, and that it can generate useful suggestions even for rare and never seen queries.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Giving suggestions to users of Web search engines is a common practice aimed at *driving* users toward the information bits they may need. Suggestions are normally provided as queries that are, to some extent, related to those recently submitted by the user. The generation process of such queries, basically, exploits the expertise of "skilled" users to help inexperienced ones. The knowledge mined for making this possible is contained in Web search engine logs which store all the past interactions of users with the search system. More the users that satisfied the same information need in the past, the more precise and effective the related suggestions provided by any query recommendation technique. On the other hand, to generate effective suggestions for user queries which are rare or have never been seen in the past is an open issue poorly addressed by state-of-the-art query suggestion techniques.

In a previous work, an interesting framework for the query suggestion problem is provided by the *Search Shortcut* model, and an evaluation metric for assessing the effectiveness of suggested queries by exploiting a query log is proposed (Baraglia et al., 2009). Basically, the model formalizes a way of exploiting the knowledge mined from query logs to help users to *rapidly* satisfy their information need. In the same work the use of *Collaborative Filtering* (CF) algorithms is investigated. However, the work highlights some limitations in the query recommendations solutions based on collaborative filtering mainly due to the poor and very sparse scoring information available in query logs. In fact, due to the long-tail distribution of query occurrences, click information for low-frequency queries is rare and very sparse. Since implicit feedback information given by

 * Corresponding author.
   E-mail address: francomaria.nardini@isti.cnr.it (F.M. Nardini).

popularity and user clicks is the only source of (positive) query scoring available, most of the queries in the query log cannot be exploited to generate the recommendation model (Adomavicius & Tuzhilin, 2005). This issue affects CF-based solutions, but also many other query recommendation techniques discussed in the literature.

In this paper we propose an efficient and effective query recommendation algorithm that can *cover* also queries in the long tail. We adopt the Search Shortcuts model and its terminology, and re-conduct the shortcut generation phase to the processing of a full-text query over an inverted file that indexes satisfactory user sessions recorded in a query log. Differently from most state-of-the art proposals, our shortcut generation algorithm aggregates implicit feedback information present in query logs at the level of single query terms, thus alleviating the data sparseness issue. The contribution of each query terms is then combined during the suggestion generation process in order to provide recommendations also for queries that are rare or even for those that were never seen in the past. Generating suggestions for rare queries is a hot research topic (Broder et al., 2009; Mei, Zhou, & Church, 2008; Song & He, 2010), and our suggestion generation technique beyond addressing the data-sparsity problem, is both very efficient and scalable, making it suitable for a large-scale deployment in real-world search engines.

Another contribution of this paper consists in the methodology adopted for manually assessing the effectiveness of query suggestion techniques. The methodology exploits the query topics and the human judgements provided by the National Institute of Standards and Technology (NIST) for running the TREC Web Track's diversity task. For the purposes of the diversity task, the NIST assessors provide 50 queries, and, for each of them, they identify a representative set of subtopics, based on information extracted from the logs of a commercial search engine. We claim that *given a query topic A with all its subtopics* $\{a_1, a_2, \ldots, a_n\}$, *and a query suggestion technique* $\mathcal{T}$, *the more the queries suggested by* $\mathcal{T}$ *for A cover the human-assessed subtopics* $\{a_1, a_2, \ldots, a_n\}$, *the more* $\mathcal{T}$ *is effective*. To assess the effectiveness of a given query suggestion technique, we thus propose to simply ask human editors to count how many subtopics are actually covered by the suggestions generated by $\mathcal{T}$ for the TREC diversity track queries. This methodology is entirely based on a publicly-available data. It can be thus considered fair and constitute a good shared base for testing and comparing query recommendation systems. We shall define the above concept better in Section 4.

The experimental evaluation conducted shows that the proposed solution outperforms remarkably two state-of-the-art algorithms chosen for performance comparison purposes (presented in Baeza-Yates & Tiberi (2007), Boldi et al. (2008) & Boldi, Bonchi, Castillo, Donato, et al. (2009)). Differently from these competitor algorithms, our solution generates in fact relevant suggestions for a vast majority of the 50 TREC queries, and the suggested queries cover a high percentage of possible subtopics. In particular, we assessed that it can generate useful suggestions even for queries that are rare or do not occur in the query log used for training the recommendation model. Moreover, the proposed algorithm outperforms the competitor solutions even on the tests measuring the effectiveness in forecasting the users' behavior recorded in historical query according to the metric used in Baraglia et al. (2009).

The main original contributions of this work are thus:

1. A novel algorithm to efficiently and effectively generate query suggestions that is robust to data sparsity.
2. A novel evaluation methodology with which we can compare the effectiveness of suggestion mechanisms.
3. An extensive evaluation comparing on the same basis the proposed solution with two state-of-the-art algorithms.

The rest of the paper is organized as follows. The next Section shows a brief overview of the state of the art in query recommendation. Section 3 briefly sketches the shortcuts model and describes the efficient algorithm designed for generating query shortcuts. The evaluation methodology based on the TREC diversification track data is discussed in Section 4 which also presents the encouraging results obtained by our solution in the performance comparisons tests conducted. Finally, Section 5 draws some conclusions and outlines future work.

2. Related work

The problem addressed in this paper is related to two related research fields that have been traditionally addressed from different points of view. We are talking about query suggestion algorithms and recommender systems.

Recommender systems are used in several domains, being specially successful in electronic commerce. They can be divided in two broad classes: those based on *content filtering*, and those on *collaborative filtering*. As the name suggests, content filtering approaches base their recommendations on the content of the items to be suggested. They face serious limitations when dealing with multimedia content and, more importantly, their suggestions are not influenced by the human-perceived *quality* of contents. On the other side, collaborative filtering solutions are based on the preferences of other users. There are two main families of collaborative filtering algorithms: memory-based and model-based. Memory-based approaches use the whole past usage data to identify similar users (Shardanand & Maes, 1995), items (Sarwar, Karypis, Konstan, & Reidl, 2001), or both (Wang, de Vries, & Reinders, 2006). Generally, memory-based algorithms are quite simple and produce good recommendations, but they usually face serious scalability problems. On the other hand, model-based algorithms construct in advance a model to represent the behavior of users, allowing to predict more efficiently their preferences. However, the model building phase can be highly time-consuming, and models are generally hard to tune, sensitive to data changes, and highly dependent on the application domain. Different approaches can be adopted based on linear algebra (Canny, 2002; Rennie &

Srebro, 2005), clustering (Ungar & Foster, 1998), latent class models (Hofmann, 2004), singular value decomposition (Paterek, 2007). An analysis of the use of collaborative filtering algorithms to the query suggestion problem can be found in Baraglia et al. (2009), where the problem descending from the poor and very sparse scoring information available in query logs is highlighted.

On the other side, query suggestion techniques address specifically the problem of recommending queries to Web search engine users, and propose specific solutions and specific evaluation metrics tailored to the Web search domain. Techniques proposed during last years are very different, yet they have in common the exploitation of usage information recorded in query logs (Silvestri, 2010). Many approaches extract the information used from the plain set of queries recorded in the log, although there are several works that take into account the chains of queries that belong to the same search session (Radlinski & Joachims, 2005). In the first category we have techniques that employ clustering algorithms to determine groups of related queries that lead users to similar documents (Baeza-Yates, Hurtado, & Mendoza, 2004; Beeferman & Berger, 2000; Wen, Nie, & Zhang, 2001). The most "representative" queries in the clusters are then returned as suggestions. Others solutions employ the reformulations of the submitted query issued by previous users (Jones, Rey, Madani, & Greiner, 2006), or propose as suggestions the frequent queries that lead in the past users to retrieve similar results (Balfe & Smyth, 2004).

Baeza-Yates and Tiberi (2007) exploit click-through data as a way to provide recommendations. The method is based on the concept of *Cover Graph*. A Cover Graph is a bipartite graph of queries and URLs, where a query q and an URL u are connected if a user issued q and clicked on u that was an answer for the query. Suggestions for a query q are thus obtained by accessing the corresponding node in the Cover Graph and by extracting the related queries sharing more URLs. The sharing of clicked URLs results to be very effective for devising related queries, and the Cover Graph solution has been chosen as one of the two query suggestion algorithms considered in this paper for experimental performance comparison.

Among the proposals exploiting the chains of queries stored in query logs, (Fonseca, Golgher, Pôssas, Ribeiro-Neto, & Ziviani, 2005) use an association rule mining algorithm to devise frequent query patterns. These patterns are inserted in a query relation graph which allows "concepts" (queries that are synonyms, specializations, generalizations, etc.) to be identified and suggested.

Boldi et al. introduce the concept of *Query Flow Graph*, an aggregated representation of the information contained in a query log (Boldi et al., 2008). A Query Flow Graph is a directed graph in which nodes are queries, and the edge connecting node q_1 to q_2 is weighted by the probability that users issue query q_2 after issuing q_1. Authors highlight the utility of the model in two concrete applications, namely, *devising logical sessions* and *generating query recommendation*. The authors refine the previous studies in Boldi, Bonchi, Castillo, Donato, and Vigna (2009) and Boldi, Bonchi, Castillo, and Vigna (2009) where a query suggestion scheme based on a random walk with restart model on the Query Flow Graph is proposed. Such suggestion algorithm is the second algorithm considered in this paper for experimental performance comparison.

A approach similar to our is represented by the query refinement/substitution technique discussed in Jones et al. (2006). The goal of query refinement is to generate a new query to replace a user's original ill-formed search query in order to enhance the relevance of retrieved results. The technique proposed includes a number of tasks such as spelling error correction, word splitting, word merging, phrase segmentation, word stemming, and acronym expansion. Our approach instead aims at suggesting users a set of queries that better specify their information needs.

The importance of rare query classification and suggestion recently attracted a lot of attention from the information retrieval community. Generating suggestions for rare queries is in fact very difficult due to the lack of information in the query logs.

Downey, Dumais, and Horvitz (2007) describe search log studies aiming at explaining behaviors associated with rare and common queries. They investigate the search behavior following the input of rare and common queries. Results show that search engines perform less well on rare queries. The authors also study transitions between rare and common queries highlighting the difference between the frequency of queries and their related information needs.

Song and He (2010) propose an optimal rare query suggestion framework by leveraging implicit feedbacks from users in the query logs. The proposed model is based on the pseudo-relevance feedback. It assumes that clicked and skipped URLs contain different level of information, and thus, they should be treated differently. Therefore, the framework optimally combines both click and skip information from users, and uses a random walk model to optimize (i) the restarting rate of the random walk, and (ii) the combination ratio of click and skip information. Experimental results on a log from a commercial search engine show the superiority of the proposed method over the traditional random walk models and pseudo-relevance feedback models.

Mei et al. propose a novel query suggestion algorithm based on ranking queries with the hitting time on a large scale bipartite graph (Mei et al., 2008). The rationale of the method is to capture semantic consistency between the suggested queries and the original query. Empirical results on a query log from a real world search engine show that hitting time is effective to generate semantically consistent query suggestions. The authors show that the proposed method and its variations are able to boost long tail queries, and personalized query suggestion.

Broder et al. propose to leverage the results from search engines as an external knowledge base for building the word features for rare queries (Broder et al., 2007). The authors train a classifier on a commercial taxonomy consisting of 6000 nodes for categorization. Results show a significant boost in term of precision with respect to the baseline query expansion methods. Lately, Broder et al. propose an efficient and effective approach for matching ads against rare queries (Broder et al., 2009). The approach builds an expanded query representation by leveraging offline processing done for related popular

queries. Experimental results show that the proposed technique significantly improves the effectiveness of advertising on rare queries with only a negligible increase in computational cost.

The idea we present in this paper follows a completely new approach. First, we infer the relevance of a query based on whether it successfully ended a search session, i.e., the *last query* of the user session allowed the user to find the information she was looking for. Recent research results have shown in fact that user behavior recorded in query log allows effective predictive models to be learned for estimating search success (Hassan, Jones, & Klinkner, 2010). *Successful sessions* have also already been taken into account as a way to evaluate promotions of search results (Smyth et al., 2005; Smyth, 2007). Similarly, *satisfactory sessions* are considered in this paper as the key factor for generating useful query recommendations. All the queries in the satisfactory sessions stored in the log which terminate with the same final query are considered "related", since it is likely that these queries were issued by different users trying to satisfy a similar information need. Thus, our technique exploit a sort of collaborative clustering of queries inferred from successful user search processes, and suggest users the final queries which are the representatives of the clusters closest to the submitted query.

3. An efficient algorithm for the query shortcuts problem

In the following we briefly recall the basis of the *Search Shortcuts Problem* (SSP) proposed in Baraglia et al. (2009), and we introduce our novel shortcuts generation algorithm.

3.1. The Search Shortcuts Problem

The SSP is formally defined as a problem related to the recommendation of queries in search engines and the potential reductions obtained in the users session length. This problem formulation allows a precise goal for query suggestion to be devised: *recommend queries that allowed "similar" users, i.e., users which in the past followed a similar search process, to successfully find the information they were looking for*. The problem has a nice parallel in computer systems: *prefetching*. Similarly to prefetching, search shortcuts anticipate requests to the search engine with suggestion of queries that a user would have likely issued at the end of her session.

We now introduce the notations and we recap the formal definition of the SSP.

Let $\mathcal{U}$ be the set of users of a Web search engine whose activities are recorded in a query log, and $\mathcal{Q}$ be the set of queries in that query log. We suppose the query log is preprocessed by using some session splitting method (e.g. Jones & Klinkner, 2008; Lucchese, Orlando, Perego, Silvestri, & Tolomei (2011)) in order to extract query *sessions*, i.e., sequences of queries which are related to the same user search task. Formally, we denote by $\mathcal{S}$ the set of all sessions in a query log. Moreover, let us denote with σ_i the *i*th query of σ. For a session σ of length n its **final query** is the query σ_n, i.e., the last query issued by the user in the session.

We say that a session σ is **satisfactory** if and only if the user has clicked on at least one link shown in the result page returned by the Web search engine for the final query σ_n, **unsatisfactory** otherwise. Finally, given a session σ of length n we denote $\sigma_{t|}$ the **head** of σ, i.e., the sequence of the first t, $t < n$, queries, and $\sigma_{|t}$ the **tail** of σ given by the sequence of the remaining $n - t$ queries.

Definition 1. We define **k-way shortcut** a function h taking as argument the head of a session $\sigma_{t|}$, and returning as result a set $h(\sigma_{t|})$ of k queries belonging to Q.

Such definition allows a simple ex-post evaluation methodology to be introduced by means of the following similarity function:

Definition 2. Given a satisfactory session $\sigma \in \mathcal{S}$ of length n, and a *k*-way shortcut function h, the similarity between $h(\sigma_{t|})$ and a tail $\sigma_{|t}$ is defined as:

$$s(h(\sigma_{t|}), \sigma_{|t}) = \frac{\sum_{q \in h(\sigma_{t|})} \sum_{m=1}^{n-t} [\![q = (\sigma_{|t})_m]\!] f(m)}{|h(\sigma_{t|})|} \tag{1}$$

where $f(m)$ is a monotonic increasing function, and function $[\![q = \sigma_m]\!] = 1$ if and only if q is equal to σ_m.

For example, to evaluate the effectiveness of a given shortcut function h, the sum (or average) of the value of s computed on all satisfactory sessions in $\mathcal{S}$ can be computed.

Definition 3. Given the set of all possible shortcut functions $\mathcal{H}$, we define **Search Shortcut Problem** (SSP) the problem of finding a function $h \in \mathcal{H}$ which maximizes the sum of the values computed by Eq. (1) on all satisfactory sessions in $\mathcal{S}$.

A difference between search shortcuts and query suggestion is actually represented by the function $[\![q = (\sigma_{|t})_m]\!]$ in Eq. (1). By relaxing the strict *equality* requirement, and by replacing it with a similarity relation – i.e. $[\![q \sim (\sigma_{|t})_m]\!] = 1$ if and only if the similarity between q and σ_m is greater than some threshold – the problem reduces, basically, to query suggestion. By defining appropriate similarity functions, the Equation in (1) can be thus used to evaluate query suggestion effectiveness as well.

Finally, we should consider the influence the function $f(m)$ has in the definition of scoring functions. Actually, depending on how *f* is chosen, different features of a shortcut generating algorithm may be tested. For instance, by setting $f(m)$ to be the

constant function $f(m) = c$, we measure simply the number of queries in common between the query shortcut set and the queries submitted by the user. A non-constant function can be used to give an higher score to queries that a user would have submitted later in the session. An exponential function $f(m) = e^m$ can be exploited instead to assign an higher score to shortcuts suggested early. Smoother f functions can be used to modulate positional effects.

3.2. The search shortcuts generation method

Inspired by the above SSP, we define a novel algorithm that aims to generate suggestions containing *only* those queries appearing as final in satisfactory sessions. The goal is to suggest queries having a high potentiality of being useful for people to reach their initial goal. As hinted by the problem definition, suggesting queries appearing as final in satisfactory sessions, in our view is a good strategy to accomplish this task. In order to validate this hypothesis, we analyzed the Microsoft RFP 2006 dataset, a query log from the MSN Search engine containing about 15 million queries sampled over one month of 2006.

First, we measured that the number of distinct queries that appear as final query in satisfactory sessions of the query log is relatively small if compared to the overall number of submitted queries: only about 10% of the total number of distinct queries in the query log occur in the last position of satisfactory user sessions. As expected, the distribution of the occurrences of such final queries in satisfactory user sessions is very skewed (as shown in Fig. 1), thus confirming once more that the set of final queries *actually* used by people is limited.

Queries which are *final* in some satisfactory sessions may obviously appear also in positions different from the last in other satisfactory sessions. We verified that, when this happens, these queries appear much more frequently in positions very close to the final one. About 60% of the distinct queries appearing in the penultimate position of satisfactory sessions are also among the final queries, about 40% in positions second to the last, 20% as third to the last, and so on. We can thus argue that *final* queries are usually *close* to the achievement of the user information goal. We consider these queries as highly valued and high quality short pieces of text expressing actual user needs.

The SSP algorithm proposed in this paper works by computing, efficiently, similarities between partial user sessions (the one currently performed) and historical satisfactory sessions recorded in a query log. Final queries of most similar satisfactory sessions are suggested to users as search shortcuts.

Let σ' be the current session performed by the user, and let us consider the sequence τ of the concatenation of all terms with possible repetitions appearing in $\sigma'_{t|}$, i.e., the head of length t of session σ'. We now compute the value of a scoring function $\delta(\tau, \sigma^s)$, which for each satisfactory session σ^s measures the similarity between its queries and the set of terms τ. Intuitively, this similarity measures how much a previously seen session overlaps with the user need expressed so far (the concatenation of terms τ serves as a bag-of-words model of user need). Sessions are ranked according to δ scores and from the subset of the top ranked sessions we suggest their final queries. It is obvious that depending on how the function δ is chosen we may have different recommendation methods. We opt for δ to be a linear combination of the BM25 metric (Robertson & Zaragoza, 2009) and the frequency of the final queries in the query log. More formally, given a satisfactory session σ^s of length n, its final query σ^s_n, and the sequence τ of the concatenation of all terms with possible repetitions appearing in $\sigma'_{t|}$, we define
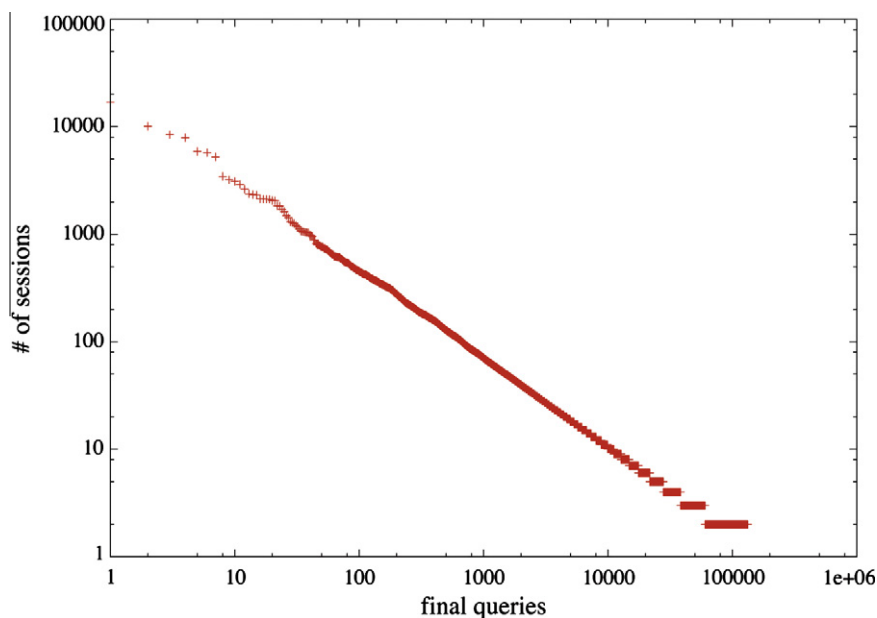

Fig. 1. Popularity of final queries in satisfactory sessions.

$$\delta(\tau, \sigma^s) = \alpha \cdot \text{BM25}(\tau, \sigma^s) + \beta \cdot \text{freq.}(\sigma_n^s)$$

We exploit an IR-like metric (BM25) because we want to take into much consideration words that are discriminant in the context of the session to which we are comparing. BM25, and other IR-related metrics, have been designed specifically to account for that property in the context of query/documents similarity. We borrow from BM25 the same attitude to adapt to this conditions. We also exploit the frequency of final queries in the scoring formula. By doing so we aim at promoting sessions containing final queries that are frequently used by users. The shortcuts generation problem has been, thus, reduced to the information retrieval task of finding highly similar sessions in response to a given sequence of queries. In our experiments we set $\alpha = \beta = 1/2$. Furthermore, we compute the similarity function δ only on the current query issued by the user instead of using the whole head of the session. We do this in order to be fair with respect to our competitors as they produce recommendations starting from a single query. We leave the study of the use of the whole head of the session for producing query recommendations as a future work.

The idea described above is thus translated into the following process (see Algorithm 1). For each unique *final query* contained in satisfactory sessions we define what we have called a *virtual document* identified by its *title* and its *content*. The title, i.e., the identifier of the document, is exactly the string of the final query. The content of the virtual document is instead composed of all the terms that have appeared in queries of all the satisfactory sessions ending with the final query. At the end of this procedure we have a set of virtual documents, one for each distinct final query occurring in some satisfactory sessions. Just to make things more clear, let us consider a toy example. Consider the two following satisfactory sessions: (*gambling*, *gambling places*, *las vegas*, *bellagio*), and (*las vegas*, *strip*, *las vegas hotels*, *bellagio*). We create the virtual document identified by the title **bellagio** and whose content is the text (*gambling gambling places las vegas las vegas strip las vegas hotels*). As you can see the virtual document actually contains also repetitions of the same term that are considered in the context of the BM25 metrics.

Algorithm 1. Offline generation of the recommendation model.

Require: a set S of user sessions recorded in the query log.
Ensure: an inverted index vd_{index} built over the virtual documents obtained from satisfactory sessions.
1: **for all** $\sigma \in S$ **do**
2:     **if** *sessionIsSatisfactory*(σ) **then**
3:         $\tau \leftarrow$ *getTermsFromSession*(σ)
4:         $vd[\sigma_n] \leftarrow \tau$ {given current satisfactory session σ, the occurrences of all the query-terms in σ are added to the (initially empty) body of the virtual document associated to final query σ_n.}
5:     **end if**
6: **end for**
7: $vd_{index} \leftarrow$ *buildInvertedIndex*(vd) {we build the inverted file indexing all the obtained virtual documents.}

All virtual documents are indexed with the preferred Information Retrieval system, and generating shortcuts for a given user session σ' becomes simply processing the query $\sigma'_{t|}$ over the inverted file indexing such virtual documents (see Algorithm 2). We know that processing queries over inverted indexes is very fast and scalable, and these important characteristics are inherited by our query suggestion technique as well.

Algorithm 2. Suggestion retrieval.

Require: the head $\sigma'_{t|}$ of length t of the current user session σ', and the recommendation model vd_{index}.
Ensure the set R of top-k scored recommendations for the given query.
1: $\tau \leftarrow$ *getTermsFromSession*$\left(\sigma'_{t|}\right)$
2: $D \leftarrow$ *getMatchingVirtualDocuments*(vd_{index}, τ)
3: $R \leftarrow$ new *heap*(k)
4: **for all** $d \in D$ **do**
5:     *shortcut* $\leftarrow$ *getTitle*(d)
6:     $R.insert$(*shortcut*, $\alpha \cdot \text{BM25}(\tau, d) + \beta \cdot freq(shortcut)$)
7: **end for**
8: **return** R.

The other important feature of our query suggestion technique is its robustness with respect to rare and singleton queries. Singleton queries account for almost 50% of the submitted queries (Silvestri, 2010), and their presence causes the issue of the sparsity of models (Adomavicius & Tuzhilin, 2005). Since we match τ with the text obtained by concatenating all the queries in each session, we are not bound to look for previously submitted queries as in the case of other suggestion algorithms.

Therefore we can generate suggestions for queries in the long tail of the distribution whose terms have some context in the query log used to build the model.

4. Assessing search shortcuts quality

The effectiveness of a query recommender systems can be evaluated by means of *user-studies* or through the adoption of some performance metrics. Unfortunately, both these methodologies may lack of generality and incur in the risk of being over-fitted on the system object of the evaluation. The evaluation methodology used in this paper tries to address pragmatically the above issues.

For what concerns the methodology based on a performance metrics, we used the one defined in Eq. (1), and we computed the average value of similarity over a set of satisfactory sessions. This performance index *objectively* measures the effectiveness of a query suggestion algorithm in foreseeing the satisfactory query for the session.

In particular, we measured the values of this performance index over suggestions generated by using our *Search Shortcuts* (SS) solution and by using in exactly the same conditions two other state-of-the-art algorithms: *Cover Graph* (CG) proposed by Baeza-Yates and Tiberi (2007), and *Query Flow Graph* (QFG), proposed by Boldi, Bonchi, Castillo, Donato, et al. (2009). These algorithms are recent and highly reputed representatives of the best practice in the field of query recommendation. To test QFG-based query suggestion we used the original implementation kindly provided us by the authors. In the case of CG, instead, we evaluate our own implementation of the technique.

For what concerns the methodology based on user-studies, we propose a approach that measures *coverage* and the *effectiveness* of suggestions against a manually assessed and publicly available dataset.

To this purpose, we exploited the query topics and the human judgements provided by NIST for running the TREC 2009 Web Track's Diversity Task (http://www.trec.nist.gov/data/web09.html). For the purposes of the TREC diversity track, NIST provided 50 queries to a group of human assessors. Assuming each TREC query as a topic, assessors were asked to identify a representative set of subtopics covering the whole spectrum of different user needs/intentions. Subtopics are based on information extracted from the logs of a commercial search engine, and are roughly balanced in terms of popularity. Obviously the queries chosen are very different and from different categories: difficult, ambiguous, and/or faceted in order to allow the overall performance of diversification methods to be evaluated and compared. Since diversity and topic coverage are key issues also for the query recommendation task (Ma, Lyu, & King, 2010), we propose to use the same third-party dataset for evaluating query suggestion effectiveness as well.

Let's now introduce the definitions of *coverage*, and *effectiveness*.

Definition 4. [Coverage] Given a query topic A with subtopics $\{a_1, a_2, \ldots, a_n\}$, and a query suggestion technique $\mathcal{T}$, we say that $\mathcal{T}$ has coverage equal to c if $n \cdot c$ subtopics match suggestions generated by $\mathcal{T}$.

In other words, a coverage of 0.8 for the top-10 suggestions generated for a query q having 5 subtopics means that 4 subtopics of q are covered by at least one suggestion.

Definition 5. [Effectiveness] Given a query topic A with subtopics $\{a_1, a_2, \ldots, a_n\}$, and a query suggestion technique $\mathcal{T}$ generating k suggestions, we say that $\mathcal{T}$ has effectiveness equal to e if $k \cdot e$ suggestions cover at least one subtopic.

In other words, an effectiveness of 0.1 on the top-10 suggestions generated for a query q means that only one suggestion is relevant for one of the subtopics of q.

The methodology just described has some net advantages. It is based on a publicly-available test collection which is provided by a well reputed third-party organization. Moreover, it grants to all the researchers the possibility of measuring the performance of their solution under exactly the same conditions, with the same dataset and the same reproducible evaluation criterium. In fact, even though the matching between suggestions and topics is still human-driven the process has a very low ambiguity as we shall discuss in the next section.

4.1. Experimental settings

The experiments were conducted using the Microsoft RFP 2006 query log which was preliminary preprocessed by converting all queries to lowercase, and by removing stop-words and punctuation/control characters.

The queries in the log were then sorted by user and timestamp, and segmented into sessions on the basis of a splitting algorithm which simply groups in the same session all the queries issued by the same users in a time span of 30 min. We tested also the session splitting technique based on the Query Flow Graph proposed in Boldi et al. (2008), but for the purpose of our technique, we did not observe a significant variation in terms of quality of the generated suggestions.

Noisy sessions, likely performed by software robots, were removed. The remaining entries correspond to approximately 9M sessions. These were split into two subsets: training set with 6M sessions and a test set with the remaining 3M sessions. The training set was used to build the recommendation models needed by CG and QFG and used for performance comparison.

Instead, to implement our SS solution we extracted satisfactory sessions present in the training set and grouped them on the basis of the final query. Then, for each distinct final query its corresponding *virtual document* was built with the
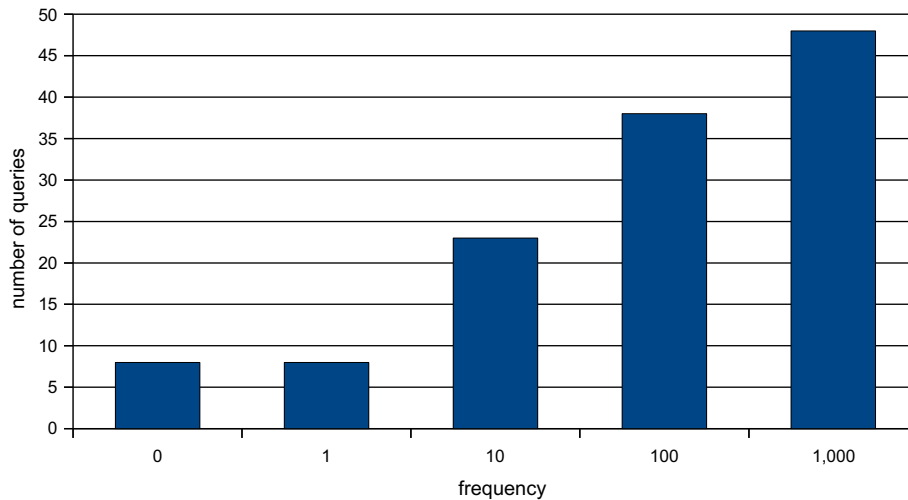
Fig. 2. Histogram showing the total number of TREC queries (on the *y* axis) having at most a certain frequency (on the *x* axis) in the training set. For instance, the third bar shows that 23 TREC queries out of 50 occur at most 10 times in the training set.

terms (with possible repetitions) belonging to all the queries of all the associated satisfactory sessions. Finally, by means of the Terrier search engine (http://www.terrier.org/), we indexed the resulting 1,191,143 virtual documents. The possibility of processing queries on such index is provided to interested readers through a simple web interface available at the address http://www.searchshortcuts.isti.cnr.it. The web-based wrapper accepts user queries, interact with Terrier to get the list of final queries (id of virtual documents) provided as top-*k* results, and retrieves and visualizes the associated query strings.

4.2. TREC queries statistics

We measured the popularity of the 50 TREC 2009 Web Track's Diversity Task queries in the training set obtained by the Microsoft RFP 2006 dataset as described in the previous section. Fig. 2 shows the cumulative frequency distribution of the 50 TREC queries. While 8/50 queries are not present in the training set, 2/50 queries occur only one time. Furthermore, 23/50 queries have a frequency lower than 10 and 33/50 queries occur lower than 100 times. The TREC dataset thus contains a valid set of queries for evaluating the effectiveness of our method as it includes several examples of unseen and rare queries, while popular queries are represented as well. Table 2 shows some queries with their popularity measured in the training set.

4.3. Search Shortcuts metric

We used Eq. (1) to measure the similarity between the suggestions generated by SS, CG, and QFG for the first queries issued by a user during a satisfactory session belonging to the test set, and the final queries actually submitted by the same user during the same session. We conducted experiments by setting the number *k* of suggestions generated to 10, and, as in Baraglia et al. (2009), we chose the exponential function $f(m) = e^m$ to assign an higher score to shortcuts suggested early. Moreover, the length *t* of the head of the session was set to $\lceil n/2 \rceil$, where *n* is the length of the session considered. Finally, the metric used to assess the similarity between two queries was the Jaccard index computed over the set of tri-grams of characters contained in the queries (Järvelin, Järvelin, & Järvelin, 2007), while the similarity threshold used was 0.9.

Due to the long execution times required by CG, and QFG for generating suggestions, it was not possible to evaluate suggestion effectiveness by processing all the satisfactory sessions in the test set. We thus considered a sample of the test set constituted by a randomly selected group of 100 satisfactory sessions having a length strictly greater than 3. The histogram in Fig. 3 plots the distribution of the number of sessions vs. the quality of the top-10 recommendations produced by the three algorithms. Results in the plot are grouped by quality range. As an example, the second group of bars shows the number of sessions for which the three algorithms generated suggestions having a quality (measured using Eq. (1)) ranging from 0.1 to 0.2. Results show that recommendations produced for the majority of sessions by QFG and CG obtains a quite low score (in the interval between 0 and 0.1), while SS produces recommendations whose quality is better distributed among all the range.

In particular, SS produces recommendations having a quality score greater than 60% for 18 sessions out of 100. Moreover, in 36 cases out of 100, SS generates useful suggestions when its competitors CG and QFG fails to produce even a single effective suggestion. On average, over the 100 sessions considered, SS obtains an average quality score equal to 0.32, while QFG and CG achieves 0.15 and 0.10, respectively.
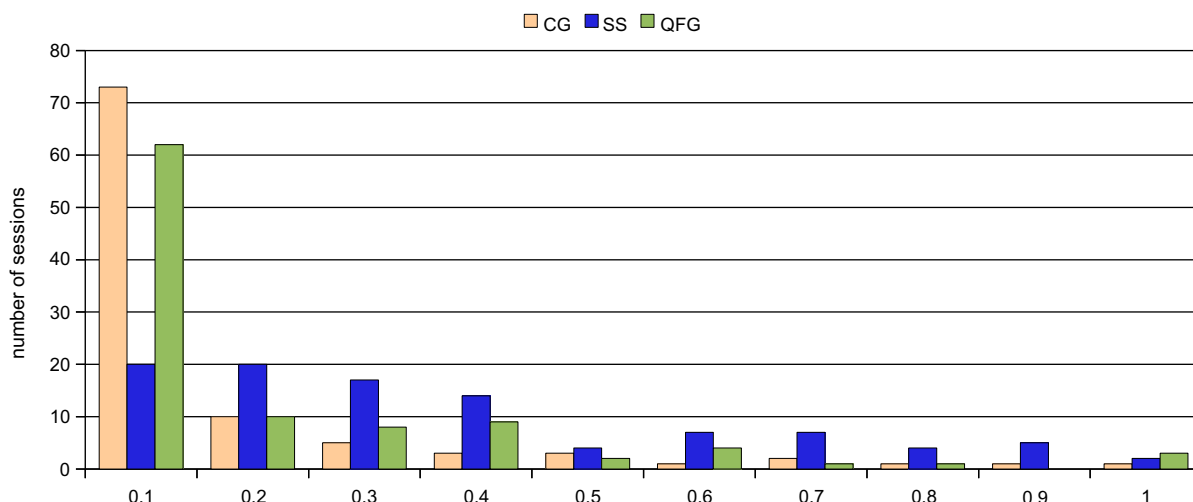
Fig. 3. Distribution of the number of sessions vs. the quality of the top-10 recommendations produced by the three algorithms.

Table 1
An example of the coverage evaluating process involving the TREC dataset. For the 8th TREC query **appraisal**, one of the assessors evaluates the coverage of suggestions generated by SS, QGF, and CG. The subtopics covered by each suggestion are reported in bold between parentheses. Suggestions not covering any of the subtopics are emphasized.

Query and its subtopics	SS	QFG	CG
TREC query (no. 8): **appraisal**	Performance appraisal (**S3**)	Online appraisals (**S4**)	Appraisersdotcom (**S4**)
S1: What companies can give an appraisal of my home's value?	Hernando county property Appraiser (**S1**)		Employee appraisals (**S3**)
			Real estate appraisals (**S1**)
S2: I'm looking for companies that appraise jewelry.	Antique appraisal (**S4**)		Appraisers (**S1**)
	Appraisers in		Employee appraisals
S3: Find examples of employee performance appraisals.	colorado (**S1**)		Forms (**S3**)
	Appraisals, etc. (**S1**)		Appraisers.com (**S4**)
S4: I'm looking for web sites that do antique appraisals.	Appraisers.com (**S4**)		*gmac*
	Find appraiser (**S1**)		Appraisers
			Beverly wv (**S1**)
	Wachovia bank		Picket fence
	Appraisals (**S1**)		Appraisal (**S1**)
	Appraisersdotcom (**S4**)		*Fossillo creek san antonio*

4.4. Suggestions quality on TREC topics

The relevance of the suggestions generated by SS, CG, and QFG w.r.t. the TREC query subtopics was assessed manually.[1] Seven volunteers were chosen among CS researchers working in different research groups of our Institute. The evaluation consisted in asking assessors to assign, for any given TREC query, the top-10 suggestions returned by SS, CG, and QFG to their related subtopic. Editors were also able to explicitly highlight that no subtopic can be associated with a particular recommendation. The evaluation process was blind, in the sense that all the suggestions produced by the three methods were presented to editors in a single, lexicographically ordered sequence where the algorithm which generated any specific suggestion was hidden. Given the limited number of queries and the precise definition of subtopics provided by NIST assessors, the task was not particularly cumbersome, and the evaluations generated by the assessors largely agree. Table 1 shows the outcome of one of the editors for the TREC query no. 8. The note in bold after each suggestion indicates the subtopic to which the particular suggestion was assigned (e.g. *employee appraisals* in the CG column matches subtopic S3). Thus for this topic this editor gave to both SS and CG a coverage of 3/4 (3 subtopics covered out of 4), while QFG was rated 1/4. Moreover, suggestions in italic, e.g. *gmac* in the CG column, were considered by the editor not relevant for any of the subtopics. Thus, for topic *appraisals* SS and QFG score an effectiveness equal to 1 (all suggestions generated were considered relevant), whereas CG score was 4/5 (2 non relevant suggestions out of 10).

---

[1] All the queries suggested by the three algorithms for the 50 TREC queries are available to the interested reader along with the associated subtopic lists at the address http://www.searchshortcuts.isti.cnr.it/TREC_results.html.

Table 2
An example of eight TREC queries with their relative frequency in the training set.

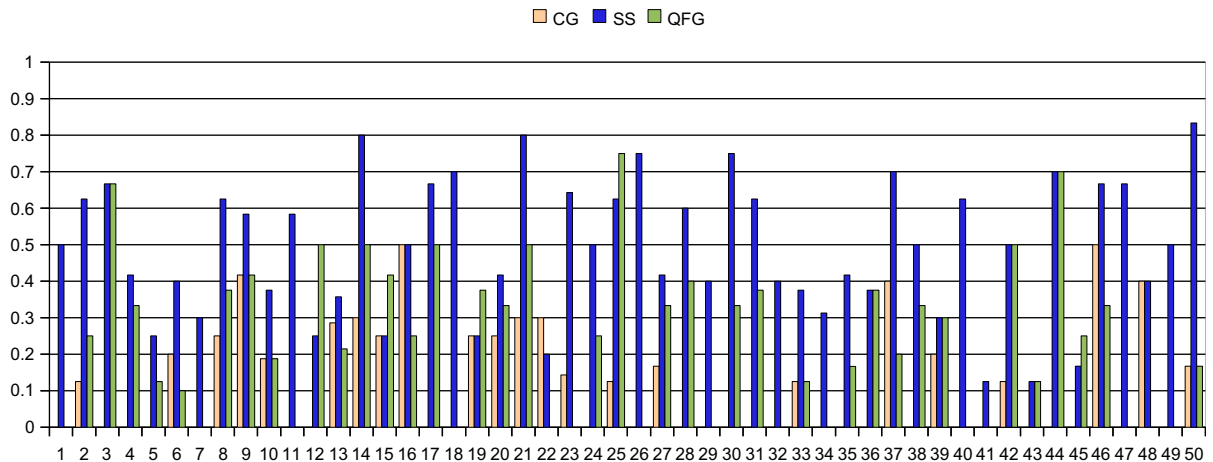| TREC query	Frequency
Wedding budget calculator	0
Flame designs	1
Dog heat	2
The music man	5
Diversity	27
Map of the united states	170
Cell phones	568
Starbucks	705

Fig. 4. Coverage of the subtopics associated with the 50 TREC diversity-track queries measured by means of an user-study on the top-10 suggestions provided by the Cover Graph (CG), Search Shortcuts (SS), and Query Flow Graph (QFG) algorithms.
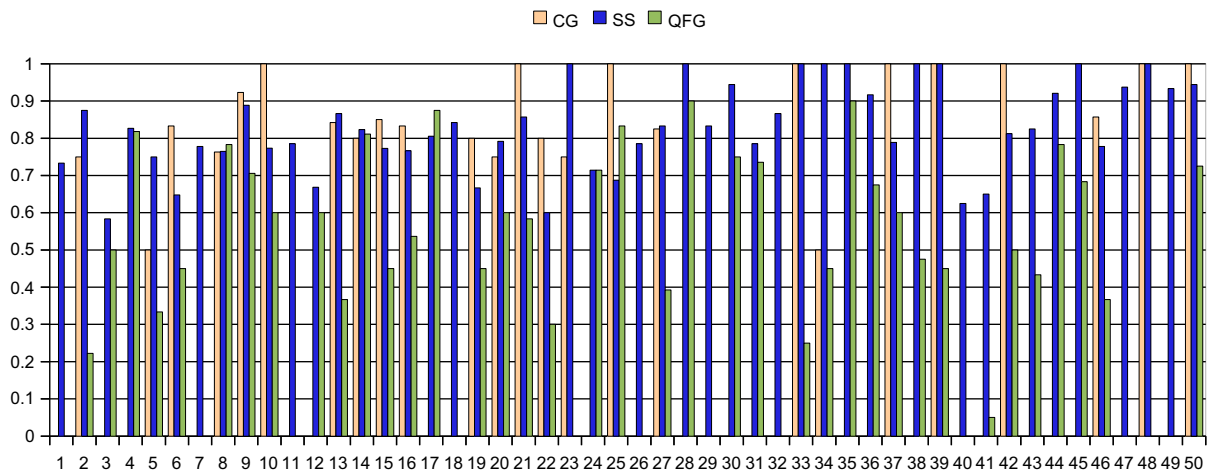

Fig. 5. Effectiveness measured on the TREC query subtopics among the top-10 suggestions returned by the Cover Graph (CG), Search Shortcuts (SS), and Query Flow Graph (QFG) algorithms.

The histogram shown in Fig. 4 plots, for each of the 50 TREC topics, the average coverage (Definition 4) of the associated subtopics measured on the basis of assessor's evaluations for the top-10 suggestions returned by SS, CG, and QFG. By looking at the Figure, we can see that SS outperforms remarkably its competitors. On 36 queries out of 50 SS was able to cover at least half of the subtopics, while CG only in two cases reached the 50% of coverage, and QFG on 8 queries out of 50. Moreover, SS covers the same number or more subtopics than its competitors in all the cases but 6. Only in 5 cases QFG outperforms SS in subtopic coverage (query topics 12, 15, 19, 25, 45), while in one case (query topic 22) CG outperforms SS. Furthermore, while
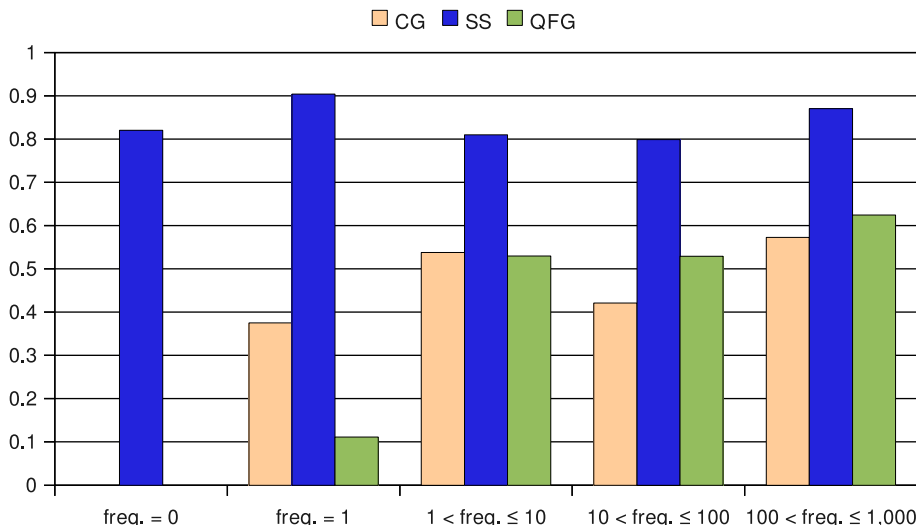
Fig. 6. Average effectiveness of the top-10 suggestions provided by the Cover Graph (CG), Search Shortcuts (SS), and Query Flow Graph (QFG) algorithms for groups of TREC queries arranged by their frequency (freq.) in the training set.

Table 3
Query suggestions provided by Search Shortcuts, Cover Graph, and Query Flow Graph for some TREC diversity-track query topics.

Queries and their subtopics	SS	CG	QFG
TREC query (no. 18): **wedding budget calculator** (faceted); Not present in the training set;			
1. I want to find online guides, tips, and checklists for planning a wedding.	Budget	*no suggestion provided*	*no suggestion provided*
2. I am looking for spreadsheets or templates to help me tabulate a budget for a wedding.	Wedding budget sheet Sample wedding budgets Budget calculator		
3. I want to find some example wedding budgets.	Budget outside wedding		
4. I'm looking for information on planning a wedding shower, like theme ideas and budget guidelines.	Wedding planning Budget wedding bouquets		
5. How can I plan an inexpensive wedding?	How to have a celebrity wedding on a budget Planning a wedding on a budget Wedding costs		
TREC query (no. 49): **flame designs** (ambiguous); Frequency in the training set: 1;			
1. Find free flame design clipart I can use on a website.	Flames	*No suggestion provided*	*No suggestion provided*
2. How do I make realistic flame images using Photoshop?	Flame fonts		
3. I'm looking for good flame tattoo designs.	Penny flame		
4. Find pictures of flames and fire.	Flame illustration		
5. I want to find flame design decals I can put on my car or motorcycle.	Flaming text Flaming fonts		
6. I'm looking for flame design stencils.	Tattoo flame designs Flame pattern comforters In flames band Flame art		
TREC query (no. 50): **dog heat** (ambiguous); Frequency in the training set: 2;			
1. What is the effect of excessive heat on dogs?	Heat	Female dogs in heat how long	Dogs in heat
2. What are symptoms of heat stroke and other heat-related illnesses in dogs?	Heat cycle of a dog Can a dog be spayed in heat	How often do dogs come in heat	Do female dogs howl in heat cycle How to tell if your
3. Find information on dogs' reproductive cycle. What does it mean when a dog is in heat?	Dogs heat Dog in heat symptoms Do female dogs howl in heat cycle Dog heat exhaustion Signs of a female dog in heat Heat cycle of dogs	Female dogs in heat	dog is pregnant Can a dog be spayed in heat Memphis spaying Pet vacs Pet vax Myspace Yspace Myspac

Table 3 (continued)

Queries and their subtopics	SS	CG	QFG
	When do dauschound go into heat		
TREC query (no. 42): *The music man* (faceted); Frequency in the training set: 5;			
1. Find lyrics for songs from The Music Man	Till there was you	The music man movie	The music man on broadway
2. Find current performances of The Music Man	Musical music man lyrics		The music man summary
3. Find recordings of songs from The Music Man	The music man soundtrack		State fair musical
4. I'm looking for the script for The Music Man	The music man summary		Till there was you
	Elephant man music		Dizzy gilelespee
	70s music rubberband man		Oysters rockefeller recipe
	Encino man songs		Archnid
	Music man lyrics		Female whale
	Free music on msn		Brewski
	Music man trouble in river city		Fats dominos first name
TREC query (n. 24): *diversity* (faceted); Frequency in the training set: 27;			
1. How is workplace diversity achieved and managed?	Diversity in education	*No suggestion provided*	Accepting diversity
2. Find free activities and materials for running a diversity training program in my office.	Diversity inclusion		Dispariging remarks
3. What is cultural diversity? What is prejudice?	Cultural diversity		Diverse world
4. Find quotes, poems, and/or artwork illustrating and promoting diversity.	Diversity test		Diversity director
	Accepting diversity		Diversity poem
	Diversity poem		Diversity test
	Diversity skills		Minority & women
	Diverse learners presentation		Civil liberties
	Picture of diverse childern		Inclusion
	Advantages of diversity		Gender and racial bias

SS is always able to cover one or some subtopics for all the cases, in 15 (27) cases for QFG (CG) the two methods are not able to cover any of the subtopics. The average fraction of subtopics covered by the three methods is: 0.49, 0.24, and 0.12 for SS, QFG, and CG, respectively.

Fig. 5 reports the effectiveness (Definition 5) of the top-10 suggestions generated by SS, QFG, and CG. Also considering this performance metric our Search Shortcuts solution results the clear winner. SS outperforms its competitors in 31 cases out of 50. The average effectiveness is 0.83, 0.43, and 0.42 for SS, QFG, and CG, respectively. The large difference measured is mainly due to the fact that both CG and QFG are not able to generate good suggestions for queries that are not popular in the training set.

Regarding this aspect, the histogram in Fig. 6 shows the average effectiveness of the top-10 suggestions returned by SS, CG and QFG measured for groups of TREC queries arranged by their frequency in the training set. SS remarkably outperforms its competitors. SS is in fact able to produce high-quality recommendations for all the categories of query analyzed, while CG and QFG can not produce recommendations for unseen queries. Furthermore, while SS produce constant quality recommendations with respect to the frequency of the TREC queries, CG and QFG show an increasing trend in the quality of recommendations as the frequency of the TREC queries increases.

For this reason, we can assert that *the SS method is very robust to data sparsity which strongly penalizes the other two algorithms, and is able to effectively produce significant suggestions also for singleton queries which were not previously submitted to the Web search engine.* We recall that singleton queries account for almost half of the whole volume of unique queries submitted to a Web search engine, and are often the hardest to answer since they ask for "rare" or badly expressed information needs. The possibility of suggesting relevant alternatives to these queries is more valuable than the one of suggesting relevant alternatives to frequent queries, which express common and often easier to satisfy needs.

Just to give an example of the results we obtained and the data on which we evaluated the quality, Table 3 reports the top-10 suggestions provided by SS, CG, and QFG for some TREC Web Track's diversity task query topics. For each query topic, the first column of the table lists the associated subtopics. These examples are representative of the figures above discussed: SS computed mostly relevant suggestions covering a significant subset of the subtopics. CG, on the other hand, performed worst and returned three suggestions only for a single query among the five reported in the table, and one single suggestion in another case. QFG returned instead 10 suggestions for three topics, and no suggestions in two cases.

5. Conclusions

We proposed a very efficient solution for generating effective suggestions to Web search engine users based on the model of *Search Shortcut*. Our original formulation of the problem allows the query suggestion generation phase to be re-conducted

to the processing of a full-text query over an inverted index. The current query issued by the user is matched over the inverted index, and final queries of the most similar satisfactory sessions are efficiently selected to be proposed to the user as query shortcuts. The way a satisfactory session is represented as a virtual document, and the IR-based technique exploited, allow our technique to generate in many cases effective suggestions even to rare or not previously seen queries. The presence of *at least* one query term in at least a satisfactory session used to build our model, permits in fact SS to be able to generate *at least* a suggestion.

By using the automatic evaluation approach based on the metric defined in Eq. (1), SS outperformed QFG in quality of a 0.17, while the improvement over CG was even greater (0.22). In 36 evaluated sessions out of 100, SS generated useful suggestions when its competitors CG and QFG failed to produce even a single useful recommendation.

An additional contribution of the paper regards the evaluation methodology used, based on a publicly-available test collection provided by a highly reputed organization such as the NIST. The proposed methodology is objective and very general, and, if accepted in the query recommendation scientific community, it would grant researchers the possibility of measuring the performance of their solution under exactly the same conditions, with the same dataset and the same evaluation criterium.

On the basis of the evaluation conducted by means of the user-study, SS remarkably outperformed both QFG and CG in almost all the tests conducted. In particular, suggestions generated by SS covered the same number or more TREC subtopics than its two counterparts in 44 cases out of 50. In 36 cases the number of subtopics covered by SS suggestions was strictly greater. Only in 5 cases QFG outperformed SS, while this never happens with CG. Also when considering effectiveness, i.e. the number of relevant suggestions among the top-10 returned, SS resulted the clear winner with an average number of relevant suggestions equal to 8.3, versus 4.3, and 4.2 for QFG, and CG, respectively. Moreover, differently from competitors SS resulted to be very robust w.r.t. data sparsity, and can produce relevant suggestions also to queries which are rare or not present in the query log used for training.

All the queries suggested by the three algorithms for the 50 TREC queries given as input to assessors are available along with the associated subtopic lists at http://www.searchshortcuts.isti.cnr.it/TREC_results.html. Moreover, a simple web-based wrapper that accepts user queries and computes the associated top-20 SS recommendations is available at http://www.searchshortcuts.isti.cnr.it.

As future works we intend to evaluate the use the whole head of the user session for producing query recommendations. Furthermore, we want to study if the sharing of the same final queries induces a sort of "clustering" of the queries composing the satisfactory user sessions. By studying such relation which is at the basis of our query shortcut implementation, we could probably find ways to improve our methodology. Finally, it would be interesting to investigate how IR-like diversification algorithms (e.g., Agraiwal, Gollapudi, Halverson, & Ieong, 2009) could be integrated in our query suggestion technique in order to obtain diversified query suggestions (Bordino, Castillo, Donato, & Gionis, 2010; Ma et al., 2010).

Acknowledgments

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE, 17*(6), 734–749.

Agraiwal, R., Gollapudi, S., Halverson, A., & Ieong, S. (2009). Diversifying search results. In *Proc. WSDM'09*. ACM.

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2004). *Query Recommendation Using Query Logs in Search Engines. LNCS* (Vol. 3268/2004). Berlin/Heidelberg: Springer.

Baeza-Yates, R., & Tiberi, A. (2007). Extracting semantic relations from query logs. In *Proc. KDD'07*. ACM.

Balfe, E., & Smyth, B. (2004). Improving web search through collaborative query recommendation. In *Proc. ECAI'04*. IOS Press.

Baraglia, R., Cacheda, F., Carneiro, V., Fernandez, D., Formoso, V., Perego, R., et al (2009). Search shortcuts: A new approach to the recommendation of queries. In *Proc. RecSys'09*. ACM.

Beeferman, D., & Berger, A. (2000). Agglomerative clustering of a search engine query log. In *Proc. KDD'00*. ACM.

Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., & Vigna, S. (2008). The query-flow graph: Model and applications. In *Proc. CIKM'08*. ACM.

Boldi, P., Bonchi, F., Castillo, C., Donato, D., & Vigna, S. (2009). Query suggestions using query-flow graphs. In *Proc. WSCD'09*. ACM.

Boldi, P., Bonchi, F., Castillo, C., & Vigna, S. (2009). From 'dango' to 'japanese cakes': Query reformulation models and patterns. In *Proc. WI'09*. IEEE.

Bordino, I., Castillo, C., Donato, D., & Gionis, A. (2010). Query similarity by projecting the query-flow graph. In *Proc. SIGIR'10*. ACM.

Broder, A., Ciccolo, P., Gabrilovich, E., Josifovski, V., Metzler, D., Riedel, L., et al (2009). Online expansion of rare queries for sponsored search. In *Proc. WWW'09*. ACM.

Broder, A. Z., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V., & Zhang, T. (2007). Robust classification of rare queries using web knowledge. In *Proc. SIGIR'07*. ACM.

Canny, J. (2002). Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR 2002 conference* (pp. 238–245). New York, NY, USA: ACM.

Downey, D., Dumais, S., & Horvitz, E. (2007). Heads and tails: Studies of web search with common and rare queries. In *Proc. SIGIR'07*. ACM.

Fonseca, B. M., Golgher, P., Pôssas, B., Ribeiro-Neto, B., & Ziviani, N. (2005). Concept-based interactive query expansion. In *Proc. CIKM'05*. ACM.

Hassan, A., Jones, R., & Klinkner, K. L. (2010). Beyond DCG: User behavior as a predictor of a successful search. In *Proc. WSDM'10* (pp. 221–230). New York, NY, USA: ACM.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst., 22*, 89–115.

Järvelin, A., Järvelin, A., & Järvelin, K. (2007). s-grams: Defining generalized n-grams for information retrieval. *IPM, 43*(4), 1005–1019.

Jones, R., & Klinkner, K. L. (2008). Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *Proc. CIKM'08*. ACM.

Jones, R., Rey, B., Madani, O., & Greiner, W. (2006). Generating query substitutions. In *Proc. WWW'06*. ACM.

Lucchese, C., Orlando, S., Perego, R., Silvestri, F., & Tolomei, G. (2011). Identifying task-based sessions in search engine query logs. In *Proc. WSDM'11* (pp. 277–286). New York, NY, USA: ACM.

Ma, H., Lyu, M.R., & King, I. (2010). Diversifying query suggestion results. In *Proc. AAAI'10*. AAAI.

Mei, Q., Zhou, D., & Church, K. (2008). Query suggestion using hitting time. In *Proc. CIKM'08*. ACM.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop* (Vol. 2007). Citeseer.

Radlinski, F., & Joachims, T. (2005). Query chains: Learning to rank from implicit feedback. In *Proc. KDD'05*. ACM Press.

Rennie, J. D. M., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd ICML 2005 conference* (pp. 713–719). New York, NY, USA: ACM.

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval, 3*(4), 333–389.

Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proc. WWW'01*. ACM.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proc. SIGCHI'95*. ACM.

Silvestri, F. (2010). Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval, 1*(1–2), 1–174.

Smyth, B. (2007). A community-based approach to personalizing web search. *Computer, 40*(8), 42–50.

Smyth, B., Balfe, E., Boydell, O., Bradley, K., Briggs, P., Coyle, M., & Freyne, J. (2005). A live-user evaluation of collaborative web search. In *IJCAI*.

Song, Y., & He, L.-W. (2010). Optimal rare query suggestion with implicit user feedback. In *Proc. WWW'10*. ACM.

Ungar, L., & Foster, D. (1998). Clustering methods for collaborative filtering. In *Proceedings of the workshop on recommendation systems*. Menlo Park, California: AAAI Press.

Wang, J., de Vries, A. P., & Reinders, M. J. T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proc. SIGIR'06*. ACM.

Wen, J.-R., Nie, J.-Y., & Zhang, H.-J. (2001). Clustering user queries of a search engine. In *Proc. WWW'01*. ACM.