Refreshing Models to Provide Timely Query Recommendations

Daniele Broccolo, Franco Maria Nardini, Raffaele Perego, Fabrizio Silvestri

ISTI - CNR
Pisa, Italy
{name.surname}@isti.cnr.it

ABSTRACT

In this work we propose a comparative study of the effects of a continuous model update on the effectiveness of well-known query recommendation algorithms. In their original formulation, these algorithms use static (i.e. pre-computed) models to generate recommendations. We extend these algorithms to generate suggestions using: a static model (no updates), a model updated periodically, and a model continuously updating (i.e. each time a query is submitted). We assess the results by previously proposed evaluation metrics and we show that the use of periodical and continuous updates of the model used for recommending queries provides better recommendations.

1. INTRODUCTION

The ocean of data on the web is continuously growing in size. Due to this reason web search engines are one of today's most used online applications to find what users need. According to Nielsen Online in October 2008 Google and Yahoo! answered more than 6 billions user searches in the US. In the latest years, web search engines have started to provide users with query recommendations to help them in refining queries and to quickly satisfy their needs. Query recommendation techniques are based on the knowledge about the behavior of past users of the search engine recorded in query logs. Basically, the behavior of many individuals is smarter than the behavior of few intelligent people.

We propose a new class of query recommender algorithms that we name "*incremental*" query recommender systems. These kind of systems update the model on which recommendations are drawn without the need for rebuilding it from scratch. That is, at regular intervals the recommender system updates the model on which suggestions are computed. In particular, we study a class of incremental recommenders where the model is updated for each received query. We study the effect on the performance (in terms of quality) of query recommender systems when varying the update interval. To do so, we propose an automatic evaluation mechanism to assess the effectiveness of query recommendation algorithms.

In this paper we aim at showing a novel class of query recommendation algorithms whose models are periodically updated as queries are submitted by users, and a comparison of four different query recommenders using new metrics. Due to space constraints we present a shortened version of our ongoing work.

Appears in the Proceedings of the 1st Italian Information Retrieval Workshop (IIR'10), January 27–28, 2010, Padova, Italy.
http://ims.dei.unipd.it/websites/iir10/index.html
Copyright owned by the authors.

2. STATIC MODELS

To validate our hypothesis about the effects of continuous model updates on query recommender systems, we consider two well-known query recommendation algorithms and we define two new algorithms in order to continuously update the model on which recommendations are computed. The first one uses association rules for generating recommendations [3] (henceforth *AssociationRules*) while the second one uses click-though data [1] (henceforth *CoverGraph*). Hereinafter, we will refer to the original formulation of the two algorithms as "*static*", as opposed to the incremental version which will be called "*incremental*".

AssociationRules. Fonseca et al. uses association rules as a basis for generating recommendations [3]. The algorithm is based on two main phases. The first one uses query log analysis for session extraction, and the second one basically extracts association rules and identifies related queries. Each session is identified by all queries sent by an user in a specific time interval ($t = 10$ minutes). The problem of mining associations is to generate all the rules having a support greater than a specified minimum threshold (*minsup*). The rationale is that if distinct queries occurs simultaneously in many user sessions then those queries are considered to be related. Suggestions for a query q are simply computed by accessing the list of rules and by suggesting the q''s corresponding to rules with the higher support values.

CoverGraph. Baeza-Yates et al. use click-through data as a way to provide recommendations [1]. The method is based on the concept of *cover graph*. A *cover graph* is a bipartite graph of queries and URLs, where a query and an URL are connected if a user clicked in a URL that was an answer for a query. To catch the relations between queries, a graph is built out of a vectorial representation for queries. Each component of the vector is weighted according to the number of times the corresponding URL has been clicked on when returned for that query. Queries are then arranged as a graph with two queries being connected by an edge if and only if the two queries share a non-zero entry, that is if for two different queries the same URL received at least one click. Furthermore, edges are weighted according to the cosine similarity of the queries they connect. Suggestions for a query q are simply obtained by accessing the corresponding

node in the cover graph and extracting the queries at the end of the top scoring edges.

3. PERIODICALLY UPDATING MODELS

We argue that the use of "incremental" algorithms for query recommendation can provide better results from two main points of view: i) models age slowly (or do not age at all), and ii) they provide better recommendations for *bursty* topics [4]. However the trade offs are: the frequency of update (i.e. update frequency has to be tuned in order to maintain an high effectiveness of recommendations), and the computational cost for updating the model (the more frequently are the updates, the less responsive the recommender system). For these reasons to design a good incremental recommender algorithm is challenging.

We design two new recommender algorithms in order to allow the update of the models at regular time intervals. The two algorithms differ from the static versions by the way they manage and use data to build the model. To achieve the main challenges both the two algorithms implement LRU structures and use HashMaps to retrieve queries and links during the update phase. Doing so, a flexible, small and easy to maintain model is obtained.

4. EXPERIMENT

Assessing the effectiveness of recommender systems is a though problem. The evaluation can be made through *user-studies* and through automatic mechanisms.

We validate our proposals by means of an automatic evaluation methodology consisting in using previously proposed metrics. Due to space constraints, we show in the following experiments results with the *QueryOverlap* metric.

Let $S = \{q_1, \ldots, q_n\}$ be a user session of length n. Let $S_1 = \{q_1, \ldots, q_{\lfloor \frac{n}{2} \rfloor}\}$ be the set of queries in the first half of the session. For each $q_j \in S_1$, let $S_2 = \{q_{j+1}, \ldots, q_n\}$ be the $n - j$ most recently submitted queries in the session, and let $R_j = \{r_1, \ldots, r_m\}$ be the set of query recommendations returned for the query. We define *QueryOverlap* as:

$$QueryOverlap(q_j) = \frac{1}{K} \sum_{\substack{r_i \in R_j \\ s_k \in S_2}} [r_i = s_k] f(k)$$

where $[r_i = s_k]$ is 1 iff the i-th element of R is equal to the k-th element of S_2, and 0 otherwise. $f(k)$ is a weighting function allowing us to differentiate the importance of each recommendation depending on the position it occupies in the second part of the session and K is a normalization factor.

The most important experiments we have conducted is to measure the benefits of continuously updating models in query recommender systems. This test is conducted generating recommendations and assessing the effectiveness of query suggestions on different time slots. Here, we briefly discuss only results for the AssociationRules algorithm.

From the plots in Figure 1 it is evident that the effectiveness of the recommendations provided by both offline and online models becomes constant from a certain period of time. However the incremental versions (both quantized and continuously updating) produce sensitively better recommendations. This is due to the inclusion in the model of new and "fresher" data. Furthermore, except for an initial phase where the model is warming up, the number of useful suggestions of the continuously updating versions of

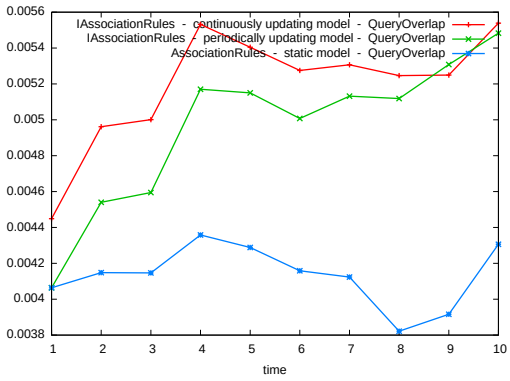

Figure 1: Comparing AssociationRules in two different implementations: static, and incremental (with periodical or continuous updates.

the algorithms is greater than the others throughout the entire observed period. The static and incrementally updating versions, indeed, produces more significant recommendations only in the very first intervals of the timeline. This is, obviously, due to both the "freshness" of the static models in the starting phases and to the cold-start problem in the continuously updating algorithms.

As a consequence of that, we prove that the aging effect on the models [2] affects the quality of the recommendations. "*Incremental*" algorithms provides a solution to this phenomenon.

5. CONCLUSIONS

In this work we propose a new class of query recommender algorithms that we name "*incremental*" query recommender systems. These kind of systems update the model on which recommendations are drawn, incrementally. In addition, we propose an automatic evaluation mechanism to assess the effectiveness of query recommendation algorithms.

Results show that continuously updating versions of the algorithms generate an higher number of useful suggestions with respect to the others throughout the entire observed period. This is a consequence of the aging effect on the models responsible for affecting the quality of the recommendations provided.

6. REFERENCES

[1] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *In Proc. KDD'07*, pages 76–85, New York, NY, USA, 2007. ACM.

[2] R. Baraglia, C. Castillo, D. Donato, F. M. Nardini, R. Perego, and F. Silvestri. Aging effects on query flow graphs for query suggestion. In *In Proc. CIKM'09.*, pages 1947–1950, New York, NY, USA, 2009. ACM.

[3] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using association rules to discover search engines related queries. In *In Proc. LA-WEB '03*, page 66, Washington, DC, USA, 2003. IEEE Computer Society.

[4] J. Kleinberg. Bursty and hierarchical structure in streams. In *In Proc. KDD'02*, pages 91–101, New York, NY, USA, 2002. ACM.