The Effects of Time on Query Flow Graph-based Models for Query Suggestion

Ranieri Baraglia
ISTI - CNR, Pisa, Italy
r.baraglia@isti.cnr.it

Carlos Castillo
Yahoo! Research Barcelona
chato@yahoo-inc.com

Debora Donato
Yahoo! Research Barcelona
debora@yahoo-inc.com

Franco Maria Nardini
ISTI - CNR, Pisa, Italy
f.nardini@isti.cnr.it

Raffaele Perego
ISTI - CNR, Pisa, Italy
r.perego@isti.cnr.it

Fabrizio Silvestri
ISTI - CNR, Pisa, Italy
f.silvestri@isti.cnr.it

ABSTRACT

A recent query-log mining approach for query recommendation is based on *Query Flow Graphs*, a markov-chain representation of the query reformulation process followed by users of Web Search Engines trying to satisfy their information needs. In this paper we aim at extending this model by providing methods for dealing with evolving data. In fact, users' interests change over time, and the knowledge extracted from query logs may suffer an aging effect as new interesting topics appear. Starting from this observation validated experimentally, we introduce a novel algorithm for updating an existing query flow graph. The proposed solution allows the recommendation model to be kept always updated without reconstructing it from scratch every time, by incrementally merging efficiently the past and present data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - *Data Mining*; H.4.3 [**Information Systems Applications**]: Communications Applications

General Terms

Algorithms

Keywords

Query Flow Graph, Query Suggestion, Topic Drift, Aging Effects, Effectiveness in Query Recommendations

1. INTRODUCTION

Web search engines are one of today's most used online applications. According to *Nielsen Online*, in October 2008 Google and Yahoo! answered more than 6 billion user searches in the US alone.

One of the main challenges for Web search engine companies is gaining user fidelity by improving their experience in the interaction with the search service. In order to pursue this goal, search engines have to figure out what users are willing to find, starting from a very terse specification of users' information need, as typical queries consists of 2 to 3 words. The vast amount of literature discussing methods for enhancing the effectiveness of search is a proof that this is far from being an easy problem.

All popular web search engines provide users with query suggestions to help them to formulate better queries and to quickly satisfy their information needs. Query recommendation techniques are typically based on the behavior of past users of the search engine, recorded in query logs. As a common practice, in fact, web search services collect detailed information about the queries of users and the URLs they click. Query suggestions take different forms: query recommendations, query expansion, query spelling correction, etc. In this paper we are interested particularly in query recommendation: the automatic generation of *interesting queries* that are related in some non trivial way to the current user information need.

A successfully query-log mining approach for generating useful query recommendation based on *Query Flow Graphs* (QFGs) [4], was recently proposed in [5]. The QFG model aggregates information in a query log by providing a markov-chain representation of the query reformulation process followed by users trying to satisfy the same information need. This paper aims at extending the QFG model by providing a methodology for dealing efficiently with evolving data. The interests of search engine users change in fact over time. New topics may suddenly become popular (this is described as a "query burst"), while others that attracted for some time the attention of users can lose importance. The knowledge extracted from query logs can thus suffer an aging effect, and the models used for recommendation rapidly becoming unable to generate useful and interesting queries. Unfortunately, building a new fresh QFG from scratch as soon as we discover the effect of aging is very expensive. We thus deal with this problem by introducing an *incremental* algorithm for updating an existing QFG. The solution proposed allows the recommendation model to be kept always updated by incrementally adding fresh knowledge and deleting the aged one.

In order to validate our claims and assess our methodology, we build different query flow graphs from the queries found on a large query log of a real-world search engine, and we analyze the quality of the recommendation model devised from these graphs to show that it inexorably ages. Then, we show that our algorithm for merging QFGs allows the recommendation model to be kept updated and we propose a general methodology for dealing with aging QFG models. Finally, we show that the computational time needed to merge QFGs is remarkably lower than the time required for building it from scratch, and we propose a distributed solution allowing to shorten further the time for the QFG creation/update.

In a previous paper it has been shown that the model built over a QFG inexorably ages over time [2]. The results we present here, are related to assess the aging effect and also to find effective anti-aging strategies to combat time effects over QFG-based models.

The paper is organized as follows. Section 2 discusses related works, while Section 3 introduces the concept of query flow graph, and provides readers with some useful notations. The data used for the experiments are described in Section 4, while their analysis finalized to the evaluation of aging effects on the recommendation models is discussed in Section 5. The incremental algorithm for updating query flow graphs with fresh data is described in Section 6. Section 7 discusses its parallel implementation. Finally, Section 8 draws some conclusions and outlines future work.

2. RELATED WORK

Different approaches have been proposed in recent years that use query logs to mine wisdom of the crowds for query suggestion.

Bruno *et al.* in [9] use an association rule mining algorithm to devise query patterns frequently co-occurring in user sessions, and a query relations graph including all the extracted patterns is built. A click-through bipartite graph is then used to identify the concepts (synonym, specialization, generalization, etc.) used to expand the original query.

Jones *et al.* in [11] introduce the notion of query substitution or query rewriting, and propose a solution for sponsored search. Such solution relies on the fact that in about half sessions the user modifies a query with another which is closely related. Such pairs of reformulated queries are mined from the log and used for query suggestion.

Baeza-Yates *et al.* [1] use a k-means algorithm to cluster queries by considering both topics and text from clicked URLs. Then the cluster most similar to user query is identified, and the queries in the cluster with the highest similarity and attractiveness (i.e. how much the answers of the query have attracted the attention of past users) are suggested. The solution is evaluated by using a query log containing only 6,042 unique queries.

Beeferman and Berger [3] apply a hierarchical agglomerative clustering technique to click-through data to find clusters of similar queries and similar URLs in a Lycos log. A bipartite graph is created from queries and related URLs which is iteratively clustered by choosing at each iteration the two pairs of most similar queries and URLs.

QFGs were introduced by Boldi *et al.* [4]. A QFG is an aggregated representation of the interesting information contained in query logs. Authors define a QFG as a directed graph in which nodes are queries, and edges are weighted by the probability of being traversed. Authors propose two weighting schemes. The first one represents the probability that two queries are part of the same search mission given that they appear in the same session, and the other one represents the probability that query q_j follows query q_i. Authors show the utility of the model in two concrete applications, namely, *finding logical sessions* and *query recommendation*. Boldi *et al.* in [5], [6] refine the previous study and propose a query suggestion scheme based on a random walk with restart model. The query recommendation process is based on reformulations of search mission. Each reformulation is classified into *query reformulation types*. Authors use four main reformulations: *generalization*, *specialization*, *error correction*, and *parallel move*. An automatic classifier was trained on manually human-labeled query log data to automatically classify reformulations. Authors showed improvements on the recommendations based on QFG models.

3. THE QUERY FLOW GRAPH

A *Query Flow Graph* is a compact representation of the information contained in a query log. It has been applied successfully to model user interactions with a web search engine and for a number of practical applications as segmenting physical sessions into logical sessions [4] or query recommendation [4], [5].

As presented in [4] a *Query Flow Graph* is a directed graph $G = (V, E, w)$ where:

- $V = Q \cup \{s, t\}$, is the set of distinct queries Q submitted to the search engine enriched with two special nodes s and t, representing a *starting state* and a *terminal state* which can be seen as the begin and the end of all the chains;
- $E \subseteq V \times V$ is the set of *directed edges*;
- $w : E \to (0..1]$ is a weighting function that assigns to every pair of queries $(q, q') \in E$ a weight $w(q, q')$.

Each distinct query is represented by a single node independently of its frequency, and the number of users who issued it. In order to build the *QFG* representing a given query log, we need to preprocess the data, sorting the queries by userid and by timestamp, and splitting them into physical sessions using a fixed time interval. In a second step we connect two queries q, q' with an edge if there is at least one session of the query log in which q and q' are consecutive. The third step of the *QFG* construction consists in weighting directed edges (q, q') on the basis of a function $w : E \to (0..1]$ that measures the probability of transition from query q to query q'. In [4], two weighting schemes are proposed. A first one based on *chaining probability* and the second one based on *relative frequencies*. For edge weighting we adopted the *chaining probability* scheme. To estimate such chaining probability, we extract for each edge (q, q') a set of features aggregated over all sessions that contain the queries q and q' appearing consecutively.

This classification step produces a set of so called *chain graphs*. Each chain graph is represented by a set of queries (i.e. nodes) interconnected by edges weighted by the probability of moving from a query to another. *Noisy* edges (i.e. those edges having a low probability of being traversed) are removed on the basis of a filtering process by means of a threshold value t.

4. EXPERIMENTAL FRAMEWORK

Our experiments have been conducted on the AOL query log and on a hardware consisting of a cluster of machines equipped with G5 PowerPCs and 5 Gb of RAM each.

The AOL data-set contains about 20 million queries issued by about $650,000$ different users, submitted to the AOL search portal over a period of three months from 1st March, 2006 to 31st May, 2006. Each query record comes with the user ID, timestamp, and the list of results returned to the user. After the controversial discussion followed to its initial public delivery, AOL has withdrawn the query log from their servers and is not offering it for download anymore. We decided to run experiments on that log anyways, because of the following reasons. First of all, the log spans a long period of time and this allows us to show how models for query suggestions degrade in a more realistic way. Second, we are not disclosing any sensitive information neither about users nor about usage behavior. Therefore, we are not breaching into the privacy of any specific user. Last, but not least, the query log is still available on the web. Everybody can easily find and download it. Indeed we consider this a strong point in favor of its use: the availability of data allows the repeatability of experiments which is an important requirement for any scientific work.

To assess the aging effects on QFG models we conducted several experiments to evaluate the impact of different factors. The log has been split into three different *segments*. Two of them have been used for training and the third one for testing. The three segments correspond to the three different months of users activities recorded in the query log. We fixed the test set – i.e. the set of queries from which we generate recommendations – to be the queries submitted in the last month. We also have conducted experiments with different training granularities, based on weekly and bi-weekly training sets. Results on those shorter training segments are consistent with those presented in the following, and we are omitting them for brevity.

The QFGs over the two monthly training segments have been constructed according to the algorithm presented by Boldi *et al.* in [4]. This method uses *chaining probabilities* measured by means of a machine learning method. The initial step was thus to extract those features from each training log, and storing them into a compressed graph representation. In particular we extracted 25 different features (time-related, session and textual features) for each pair of queries (q, q') that are consecutive in at least one session of the query log.

Table 1 shows the number of nodes and edges of the different graphs corresponding to each query log segment used for training.

time window	id	nodes	edges
March 06	$\mathcal{M}_1$	3,814,748	6,129,629
April 06	$\mathcal{M}_2$	3,832,973	6,266,648

Table 1: Number of nodes and edges for the graphs corresponding to the two different training segments.

It is important to remark that we have not re-trained the classification model for the assignment of weights associated with QFG edges. We reuse the one that has been used in [4] for segmenting users sessions into query chains[1]. This is another point in favor of QFG-based models. Once you train the classifier to assign weights to QFG edges, you can reuse it on different data-sets without loosing in effectiveness. We want to point out, indeed, that what we are evaluating in this work is that QFGs themselves age much faster than the model used to build them. This is a subtle difference we want to state clearly.

[1]We thank the authors of [4] for providing us their model.

Once the QFG has been built, the query recommendation methods are based on the probability of being at a certain node after performing a random walk over the query graph. This random walk starts at the node corresponding to the query for which we want to generate a suggestion. At each step, the random walker either remains in the same node with a probability α, or it follows one of the out-links with probability equal to $1 - \alpha$; in the latter case, out-links are followed proportionally to $w(i, j)$. In all the experiments we computed the stable vector of the random walk on each QFG by using $\alpha = 0.15$. Actually, the stable vector is computed according to a Random Walk with Restart model [13]. Instead of restarting the random walk from a query chosen uniformly at random, we restart the random walk only from a given set of nodes. This is done by using a preference vector v, much in the spirit of the Topic-based PageRank computation [10], defined as follows. Let $q_1, \ldots, q_n$ be a query chain (q_1 is the most recently submitted query). The preference vector v is defined in the following way: $v_q = 0$ for all $q \notin q_1, \ldots, q_n$ and $v_{q_i} \propto \beta^i$. β is a weighting factor that we set in all of our experiments to be $\beta = 0.90$.

5. EVALUATING THE AGING EFFECT

One of the main goals of this paper is to show that time has some negative effects on the quality of query suggestions generated by QFG-based models. It is also worth remarking that we can safely extend the discussion that follows also to suggestion models different from QFG-based ones. As a matter of fact, the presence of "*bursty*" [12] topics could require frequent model updates whatever model we are using. To validate our hypothesis about the aging of QFG-based models we have conducted experiments on models built on the two different training segments described in the above section.

In order to assess the various reasons why a QFG-based model ages we have considered, for each segment, two classes of queries, namely $\mathcal{F}_1$, and $\mathcal{F}_3$, which respectively correspond to queries having a strong decrease and a strong increase in frequency. $\mathcal{F}_1$ is the set of the 30 queries that are among the 1,000 most frequent queries in the first month ($\mathcal{M}_1$) but whose frequency has had the greater drop in the last month covered by the query log ($\mathcal{M}_3$). Conversely, $\mathcal{F}_3$ is the set of the 30 queries among the 1,000 most frequent queries in the test log $\mathcal{M}_3$ whose frequency has the greater drop in the first part of the log $\mathcal{M}_1$. Actually, to make the assessment more significant, we do not include queries that are too similar, and we do not include queries containing domain names within the query string. Figure 1 graphically show where the selected queries for each class fall when we plot the popularity of the top-1000 most frequent queries in $\mathcal{M}_1$ ($\mathcal{M}_3$) by considering query ids assigned according to frequencies in $\mathcal{M}_3$ ($\mathcal{M}_1$).

Some examples of queries in $\mathcal{F}_1$ are: "*shakira*", "*americanidol*", "*ecards*", "*nfl*". Such queries are related to particular events in March 2006, for instance singer Shakira in March 2006 released a new album. Some examples of queries in $\mathcal{F}_3$ are: "*mothers day gift*", "*mothers day poems*", "*memorial day*", "*da vinci code*". As in the previous case, $\mathcal{F}_3$ queries are strongly related to that particular period of

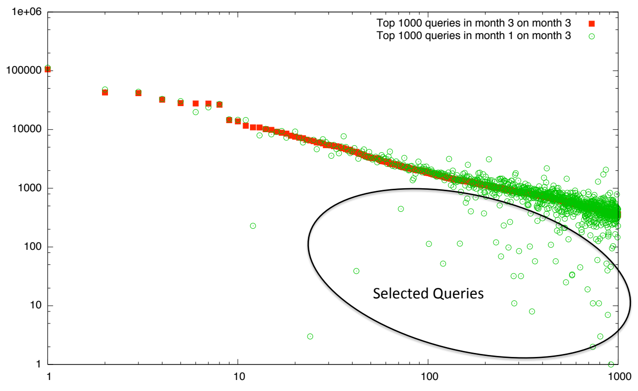

Figure 1: Queries in $\mathcal{F}_3$. The set of top 1,000 queries in $\mathcal{M}_3$ compared with the same set projected on $\mathcal{M}_1$. Query identifiers are assigned according to frequencies in $\mathcal{M}_3$. The circled area in the plot highlights the zone from where $\mathcal{F}_3$ was drawn.

time. For instance, in May 2006 the movie adaptation of the popular book "Da Vinci Code" was released.

We selected two distinct sets because we want to assess the effectiveness of recommendations for both new or emerging query topics in the test log (i.e. queries in $\mathcal{F}_3$), and for queries that are frequent in the first month but poorly represented (or absent) in the test month (i.e. queries in $\mathcal{F}_1$).

The first evaluation we perform is a *human-based assessment* of the quality of query suggestions generated by models trained on the two different segments. From each query in $\mathcal{F}_1$ and $\mathcal{F}_3$ we generated the top 20 recommendations using four different sets of QFG-based models: three of them are filtered with different values of the threshold t (0.5, 0.65, and 0.75), one is generated without filtering ($t = 0$). Each set consists of QFGs built on either $\mathcal{M}_1$ or $\mathcal{M}_2$.

The generated recommendations were manually evaluated and classified as **useful** and **not useful**. We consider useful a recommendation that undoubtedly interprets the possible intent of the user better than the original query.

filtering threshold	average number of useful suggestions on $\mathcal{M}_1$			average number of useful suggestions on $\mathcal{M}_2$		
	$\mathcal{F}_1$	$\mathcal{F}_3$	$\mathcal{F}_1 \cup \mathcal{F}_3$	$\mathcal{F}_1$	$\mathcal{F}_3$	$\mathcal{F}_1 \cup \mathcal{F}_3$
0	2.51	2.02	2.26	2.12	2.46	2.29
0.5	3.11	2.69	2.9	2.88	2.87	2.87
0.65	3.02	2.66	2.84	2.8	2.71	2.76
0.75	3	2.64	2.82	2.72	2.68	2.7

Table 3: Model aging statistics varying the model type and the temporal window. Results were manually assessed.

Table 3 shows the results of the human assessment performed by counting, for each query and the three different threshold levels, the number of useful suggestions. We averaged the counts over all the queries evaluated. For each training period we show the average number of useful suggestion for queries in the three different groups, i.e. $\mathcal{F}_1$, $\mathcal{F}_3$, and $\mathcal{F}_1 \cup \mathcal{F}_3$.

From the table we can draw some interesting conclusions. First, the performance of the models built from $\mathcal{M}_1$ and $\mathcal{M}_2$ are quite similar (column $\mathcal{F}_1 \cup \mathcal{F}_3$). This might seem a counterexample to the hypothesis that the models age. Actually, by breaking down the overall figure into separate figures for $\mathcal{F}_1$ and $\mathcal{F}_3$ we can observe that for all the queries in $\mathcal{F}_3$ the suggestions built from $\mathcal{M}_2$ are more useful than those built on $\mathcal{M}_1$.

Furthermore, by inspecting some of the suggestions generated for the queries shown in Table 2, it is evident that some of the suggestions are "*fresher*" (i.e. more up-to-date) in the case of a model built on $\mathcal{M}_2$ than those obtained on models built on $\mathcal{M}_1$. This is particularly true for queries in $\mathcal{F}_3$. For instance, for the query "*lost*" suggestions computed by a model trained on $\mathcal{M}_2$ appear to be more meaningful than those suggested using an old model on that particular period of time. Furthermore, another interesting observation is that filtering (i.e. removing noisy edges) works pretty well since it increases the average number of useful suggestions.

When we performed the assessment of the suggestions we noted a phenomenon regarding the scores computed on the different QFGs by the random walk-based method. Let us consider again the results shown in Table 2 and let us look at the suggestions, with the relative scores, computed for 6 queries (3 queries from $\mathcal{F}_1$ and 3 queries from $\mathcal{F}_3$) on $\mathcal{M}_1$ and $\mathcal{M}_2$.

As we go further down the list sorted by score, when the quality of the suggestions starts to degrade, we often observe that the useless suggestions are associated with the same low score values, e.g. "*regions banking*", "*aol email only*", "*adultactioncamcom*" are three different (and useless) query suggestions for the query "*harley davidson*" whose QFG computed score is always 1394.

From the above observation we make the following hypothesis that we will use to derive a second automatic evaluation methodology to assess the "usefulness" of suggestions:

> *when a QFG-based query recommender system gives the same score to consecutive suggestions, these recommendations and the following ones having a lower score are very likely to be useless.*

A QFG-based recommender system recommends queries by computing a random walk with restart on the model. At each step, the random walker either remains in the same node with a probability α, or it follows one of the out-links with probability equal to $1 - \alpha$. Out-links are followed proportionally to $w(i, j)$. Let us suppose the recommender system starts recommending more than k queries sharing the same score for the given query q. On the QFG model it means that the query q has more than k out-links sharing the same probability ($w(i, j)$). Due to the lack of information the system is not able to assign a priority to the k recommended queries. This is the reason why we consider these recommendations as "useless".

This heuristic considers useful k query recommendations if the suggestions following the top-k recommended queries have equal scores associated with them. Consider again the case of the query *harley davidson*, we have six queries with different scores and then the remaining queries (for which the associated scores are equal) are clearly useless.

We perform the automatic analysis described above to the 400 most frequent queries in the third month for which recommendations were generated on models built on either $\mathcal{M}_1$ or $\mathcal{M}_2$. For all the experiments we set $k = 3$. Table 4 shows that according to this measure of quality filtered models works better than unfiltered ones. The filtering process

Set	Query	$\mathcal{M}_1$		$\mathcal{M}_2$	
		Score	Suggestions	Score	Suggestions
$\mathcal{F}_3$	da vinci	49743	da vinci's self portched black and white	73219	da vinci and math
		47294	the vitruvian man	33769	da vinci biography
		35362	last supper da vinci	31383	da vinci code on portrait
		31307	leonardo da vinci	29565	flying machines
		30234	post it	28432	inventions by leonardo da vinci
		30234	handshape 20stories	26003	leonardo da vinci paintings
				23343	friends church
				23343	jerry c website
	survivor	8533	watch survivor episodes	7392	survivor preview
		8083	survivor island	7298	watch survivor episodes
		4391	2006 survivor	7110	survivor island album
		4310	bubba gump hat	4578	survivor edition 2006
		4310	big shorts	3801	cbs the great race
				3801	chicken and broccoli
	lost	16522	lost fan site	5138	lost season 2
		3634	abcpreview.go.com	3742	lost update
		2341	altricious	2162	lost easter eggs
		2341	5 year treasury rate	1913	abcpreview.go.com
		2341	1-800-sos-radon	1913	antique curio cabinets
				1913	4440
$\mathcal{F}_1$	anna nicole smith	11113	anna nicole smith nude	23497	anna nicole smith recent news
		11101	anna nicole smith - supreme court	18694	anna nicole smith and supreme court
		11054	anna nicole smith diet	18546	anna nicole smith and playboy
		10274	anna nicole smith with liposuction	16836	anna nicole smith pictures
		4677	cameron diaz video	15289	anna nicole smith free nude pics
		4677	calcination	13436	anna nicole smith diet
				6642	branson tractors
				6642	bandlinoblu jeans
	harley davidson	5097	harley davidson ny	5749	harley davidson premium sound system owners manual
		2652	american harley davidson	3859	automatic motorcycles
		2615	2002 harley davidson ultra classic	3635	harley davidson credit
		2602	adamec harley davidson	3618	cherokee harley davidson
		2341	air fight	2103	harley davidson sporster
		2341	928 zip code	1965	2002 harley davidson classic
		2341	antispy ware	1394	regions banking
				1394	aol email only
				1394	adultactioncamcom
	shakira	10989	shakira video	3281	hips don't lie
		7522	shakira albums	3281	shakira hips don't lie video
		7522	shakira my hips don't lie	3175	shakira video
		5836	shakira biography	3042	shakira wallpaper
		3864	70s music funk	2811	shakira album
		3864	97.1zht	2592	shakira nude
				1868	cant fight the moonlight
				1868	free video downloads

Table 2: Some examples of recommendations generated on different QFG models. Queries used to generate recommendations are taken from different query sets. For each query we present the most important recommendations with their assigned relative scores.

reduces the "noise" on the data and generates more precise knowledge on which recommendations are computed. Furthermore, the increase is quite independent from the threshold level, i.e. by increasing the threshold from 0.5 to 0.75 the overall quality is, roughly, constant.

filtering threshold	average number of useful suggestions on $\mathcal{M}_1$	average number of useful suggestions on $\mathcal{M}_2$
0	2.84	2.91
0.5	5.85	6.23
0.65	5.85	6.23
0.75	5.85	6.18

Table 4: Recommendation statistics obtained by using the automatic evaluation method on a set of 400 queries drawn from the most frequent in the third month.

We further break down the overall results shown in Table 4 to show the number of queries on which the QFG-based model generated a given number of useful suggestions. We plot this histogram to compare those numbers on $\mathcal{M}_1$ and $\mathcal{M}_2$ in Figure 2. To highlight more the effect of incremental updates we show in Figure 3 the total number of queries having at least a certain number of useful recommendation. For example, the third bucket shows how many queries have at least three useful suggestions. For each bucket, results for $\mathcal{M}_2$ are always better than the ones for $\mathcal{M}_1$.

First of all, when we train a QFG-based model on $\mathcal{M}_2$ the percentage of queries having 0 useful results is remarkably lower than those measured on the model trained on

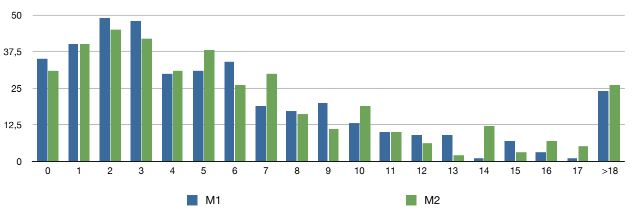

Figure 2: Histogram showing the number of queries (on the y axis) having a certain number of useful recommendations (on the x axis). Results are evaluated automatically.

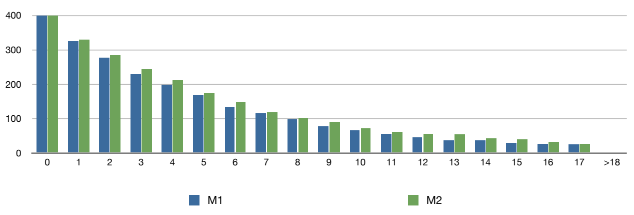

Figure 3: Histogram showing the total number of queries (on the y axis) having at least a certain number of useful recommendations (on the x axis). For instance the third bucket shows how many queries have at least three useful suggestions.

$\mathcal{M}_1$. Furthermore, for Figure 3 we can observe that a model trained on $\mathcal{M}_2$ has a larger percentage of queries for which the number of useful suggestions is at least 4.

This confirms our hypothesis that QFG-based recommendation models age and have to be updated in order to always generate useful suggestions.

6. COMBATING AGING IN QUERY-FLOW GRAPHS

Models based on Query Flow Graphs age quite rapidly in terms of their performance for generating useful recommendations.

The pool of queries we use to generate recommendation models ($\mathcal{M}_1$) contains both frequent and time-sensitive queries. We consider time-sensitive those queries that are both frequent and have a large variation with respect to their value in the previous month(s) (e.g. $> 50\%$ of positive variation). Time-sensitive queries are related for instance to movies, new products being launched, fashion, and in general with news events generating query bursts. In Figure 1 it is easy to identify time-sensitive queries by looking at those with the greater variation between their frequencies over the two months.

As Tables 3 and 8 show, if we compute recommendations on older models the average recommendation quality degrades. Indeed, "*stable*" queries, i.e. those queries that are (relatively) frequent in each period of the year, also have to be taken into account by the model. Therefore, we must be able to give suggestions that are not too much influenced by time or, in other words, to update the recommendation model instead of rebuilding a new model from scratch disregarding older knowledge.

There are two ways of building an updated QFG: i) rebuilding the model by considering the whole set of queries from scratch, or ii) using the old QFG-based model and adding fresh queries to it.

A straightforward option is to compute a new QFG representing the two months. The new part of the query log ($\mathcal{M}_2$) is merged with the old one ($\mathcal{M}_1$) obtaining a new data-set to be used for building the new model. The merged data-set can be processed from scratch by performing all the steps necessary to build a QFG (i.e. preprocessing, features generation, compression, normalization, chain graph generation, random walk computation). The old model is simply discarded.

The other option is computing recommendations on an "*incremental model*" that is built by merging the old QFGs with the one obtained from the new queries. In this way: i) we refresh the model with new data covering time-related queries; ii) the "old" part of the model contributes to maintain quality on frequent and time-unrelated queries. The methodology we are proposing incrementally extends the recommendation model with a sub-model built on more recent data. A sound incremental approach has to consistently update the old model with fresh data continuously or after fixed periods of time. Another advantage of this approach is that the new model can be built by spending only the time needed to build a QFG from a relatively small set of new queries, plus the cost of the merging process.

Let us introduce an example to show the main differences among the two approaches in terms of computational time. Table 5 shows the total elapsed times to create different QFGs. Suppose the model used to generate recommendations consists of a portion of data representing one month (for $\mathcal{M}_1$ and $\mathcal{M}_2$) or two months (for $\mathcal{M}_{12}$) of the query log. The model is being updated every 15 days (for $\mathcal{M}_1$ and $\mathcal{M}_2$) or every 30 days (for $\mathcal{M}_{12}$). By using the first approach, we pay 22 (44) minutes every 15 (30) days to rebuild the new model from scratch on a new set of data obtained from the last two months of the query log. Instead, by using the second approach, we need to pay only 15 (32) minutes for updating the one-month (two-months) QFG.

Dataset	"From scratch" strategy [min.]	"Incremental" strategy [min.]
$\mathcal{M}_1$ (March 2006)	21	14
$\mathcal{M}_2$ (April 2006)	22	15
$\mathcal{M}_{12}$ (March and April)	44	32

Table 5: Time needed to build a Query Flow Graph from scratch and using our "incremental" approach (from merging two QFG representing an half of data).

The time spent in updating incrementally the model is, in practice, shorter than the time needed to build the model from scratch (in our case it is almost two third (i.e. 32%) that time). The process of merging two QFGs can be performed using an open-source Java tool [8] that implements a graph algebra giving users the possibility to perform some operations on WebGraph [7] encoded graphs. In our experiments we used three different QFGs built on $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_{12}$ defined as in the first column of Table 5.

query	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_{12}$
mothers day	2	3	3
da vinci	4	6	7
lost	2	4	6
philippines	2	2	3

Table 6: Manual assessment of the number of useful recommendations generated for some time-related queries on the three different models.

Table 6 shows the importance of having an incremental approach for time-related queries. It is evident from the table that the model built on $\mathcal{M}_{12}$ always gives the best recommendations, in terms of quality, with respect to the two separate models $\mathcal{M}_1$ and $\mathcal{M}_2$.

Table 7 shows example query suggestions generated by the system, to demonstrate the improved quality of the recommendations. These results suggest a positive effect of the incremental-update method on the recommendation quality.

As in the previous section we evaluated the quality of the recommendations, also by using the automatic procedure above described. The results are shown on Table 8. We have used the same set of 400 queries for which recommendations were generated using the QFG built on $\mathcal{M}_{12}$.

Again, the results suggested by the anecdotal evidence, are confirmed by the assessment procedure. The model built on the two merged train segments is better than the single $\mathcal{M}_2$ model (which was, in turn, better than the model built on $\mathcal{M}_1$). Improvements, in this case, are quite significant and range from 25% to 28% in accuracy.

Using the automatic evaluation method we have investi-

Query Set	Query	$\mathcal{M}_{12}$	
$\mathcal{F}_3$	da vinci	86251	da vinci and math
		85436	da vinci biography
		83427	da vinci code on portrait
		82119	da vinci's self portched black and white
		80945	flying machines
		79563	inventions by leonardo da vinci
		74346	leonardo da vinci paintings
		30426	friends church
	survivor	7250	survivor preview
		7250	watch survivor episodes
		7236	survivor island panama
		7110	survivor island exile
		4578	survivor edition 2006
		3980	survivor games
		3717	big shorts
		3717	baltimore bulk trash
	lost	13258	lost fan site
		3716	lost season 2
		3640	abcpreview.go.com
		3640	lost chat room
		3582	lost update
		3272	lost easter eggs
		1913	henry gale
		1858	4440
		1858	1-800-sos-radon
		1858	4440
$\mathcal{F}_1$	anna nicole smith	15174	anna nicole smith recent news
		14876	anna nicole smith and supreme court
		13567	anna nicole smith court appeal
		12768	anna nicorle smith and playboy
		10509	anna nicole smith pictures
		9832	anna nicole smith free nude pics
		9411	anna nicole smith show
		8971	anna nicole smith diet
		8880	branson tractors
		8880	bandlinoblu jeans
	harley davidson	5969	harley davidson premium sound system owners manual
		4073	harley davidson ny
		4001	automatic motorcycles
		3738	cherokee harley davidson
		3038	harley davidson credit
		2562	custom harley davidson
		2494	harley davidson sporster
		2166	2002 harley davidson classic
		2085	regions banking
		2085	1998 dodge ram ground effects kits
		2085	adultactioncamcom
	shakira	4174	hips don't lie
		4174	shakira albums
		4140	shakira hips don't lie video
		3698	shakira video
		3135	shakira nude
		3099	shakira wallpaper
		3020	shakira biography
		3018	shakira aol music
		2015	free video downloads

Table 7: Some examples of recommendations generated on different QFG models. Queries used to generate recommendations are taken from different query sets.

filtering threshold	average number of useful suggestions on $\mathcal{M}_2$	average number of useful suggestions on $\mathcal{M}_{12}$
0	2.91	3.64
0.5	6.23	7.95
0.65	6.23	7.94
0.75	6.18	7.9

Table 8: Recommendation statistics obtained by using the automatic evaluation method on a relatively large set of 400 queries drawn from the most frequent in the third month.

gated the main reasons why we obtain such an improvement. Looking at the different bars in Figure 4 we can observe that values that have had the greatest improvement are those corresponding to a number of suggestions larger than 8 (with the only exceptions of the cases of 10 and 14 suggestions). In particular the improvement is remarkable in the case of "more than 18" useful suggestions given. Figure 5 also shows the total number of queries having at least a certain number of useful recommendations. Again, results for $\mathcal{M}_{12}$ are remarkably better than those referring to $\mathcal{M}_1$, and $\mathcal{M}_2$.

To conclude, we have shown that recommendations given

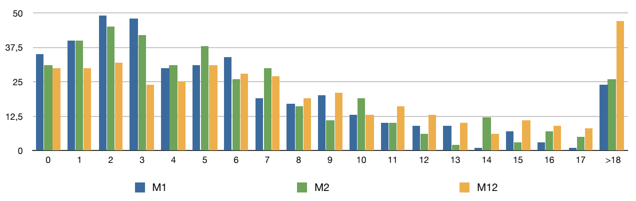

Figure 4: Histogram showing the number of queries (on the y axis) having a certain number of useful recommendations (on the x axis). Results are evaluated automatically.

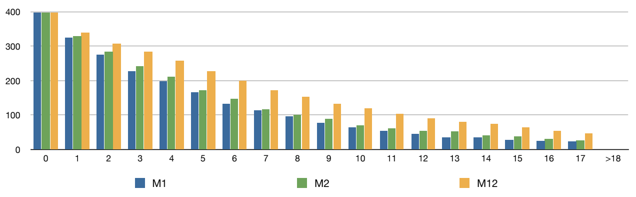

Figure 5: Histogram showing the total number of queries (on the y axis) having at least a certain number of useful recommendations (on the x axis). For instance the third bucket shows how many queries have at least three useful suggestions.

using QFG-based models are sensitive to the aging of the query base on which they are built. We have also shown the superiority of QFG-based models built on queries drawn from larger period of time and we have shown how to build such a models without the need of retraining them from scratch.

7. DISTRIBUTED QFG BUILDING

In this section we present a method to build QFGs that exploit the observation made above on the feasibility of an incremental approach to QFG-based model update. We present an approach for building a QFG-based model on a distributed architecture.

7.1 Divide-and-Conquer Approach

The first approach exploits a parallel divide-and-conquer computation that proceeds by dividing the query log into m distinct segments, building the QFG in parallel on each processor available and the iteratively merging the different segments until a single QFG is obtained.

The process is depicted in Figure 6. Our algorithm builds a QFG as follows:

1. the query log is split into m parts. In the example represented in Figure 6 files were split into 15 days intervals;

2. the features requested to build the QFG are extracted from the data contained in each interval. Each interval, is processed in parallel on the different machines of the cloud;

3. each part is compressed using the WebGraph framework, obtaining a partial data-graph;

4. using the graph algebra described in [8], each partial graph is iteratively merged. Each iteration is done in parallel on the different available nodes of the cloud;

5. the final resulting data-graph is now processed with other steps [4] (normalization, chain extraction, random walk) to obtain the complete and usable QFG.

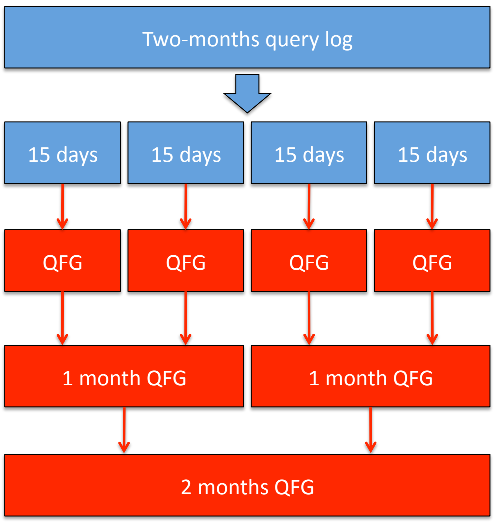

Figure 6: Example of the building of a two months query flow graph with a "parallel" approach.

Table 9 summarizes the computational costs of building a QFG in a distributed way. The main benefit of this approach is to significantly reduce the time needed to perform the preliminary generation step.

generation of 15-days data-graph	6 min 15 sec
merge of two 15-days data-graphs	5 min 23 sec
merge of two one-month data-graphs	11 min 04 sec
TOTAL	22 min 42 sec

Table 9: Time needed to build a two-months data-graph using our "incremental" approach and splitting the query log in 4 parts.

8. CONCLUSION AND FUTURE WORK

In this paper we have studied the effect of time on recommendations generated using *Query Flow Graphs* [4] (QFGs). These models aggregate information in a query log by providing a markov-chain representation of the query reformulation process followed by multiple users. In this paper we have shown how to extend QFG-based recommendation models to evolving data. We have shown that the interests of search-engine users change over time and new topics may become popular, while other that focused for some time the attention of the crowds can suddenly loose importance. The knowledge extracted from query logs can thus suffer from an aging effect, and the models used for recommendations rapidly become unable to generate useful and interesting suggestions. We have shown that the building of a new fresh QFG from scratch is expensive. To overcome this problem we have introduced an *incremental* algorithm for updating an existing QFG. The solution proposed allows the recommendation model to be kept always updated by incrementally adding fresh knowledge and deleting the aged one.

In order to validate our claims and assess our methodology, we built different QFGs from the query log of a real-world search engine, and we analyze the quality of the recommendation models obtained from these graphs to show that they inexorably age. It is worth noticing that a comprehensive user-study is (almost) impossible on this kind of task. To assess the effect of aging with a survey, we would need to make users aware of the context and news events happened in the period to which the query log is referred (March-May 2006). Then, we have proposed a general methodology for dealing with aging QFG models that allows the recommendation model to be kept up-to-dated in a remarkably lower time than that required for building a new model from scratch. As a side result we have proposed a parallel/distributed solution allowing to make QFG creation/update operations scalable.

9. REFERENCES

[1] R. Baeza-Yates, C. Hurtado, and M. Mendoza. *Query Recommendation Using Query Logs in Search Engines*, volume 3268/2004 of *LNCS*. Springer, 2004.

[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 *Proc. CIKM'09*. ACM, 2009.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proc. KDD'00*. ACM, 2000.

[4] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *Proc. CIKM'08*. ACM, 2008.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *Proc. WSCD'09*. ACM, 2009.

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

[7] P. Boldi and S. Vigna. The webgraph framework i: compression techniques. In *Proc. WWW'04*. ACM Press, 2004.

[8] I. Bordino and D. Laguardia. Algebra for the joint mining of query log graphs. http://www.dis.uniroma1.it/∼bordino/qlogs_algebra/, 2008.

[9] B. M. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani. Concept-based interactive query expansion. In *Proc. CIKM'05*. ACM, 2005.

[10] T. H. Haveliwala. Topic-sensitive pagerank. In *Proc. WWW'02*. ACM, 2002.

[11] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. WWW'06*. ACM Press, 2006.

[12] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proc. KDD'02*. ACM, 2002.

[13] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *Proc. ICDM'06*. IEEE CS, 2006.