Query Similarity by Projecting the Query-Flow Graph

Ilaria Bordino[1]*
bordino@dis.uniroma.it

Carlos Castillo[2]
chato@yahoo-inc.com

Debora Donato[2]
debora@yahoo-inc.com

Aristides Gionis[2]
gionis@yahoo-inc.com

[1]DIS, Sapienza Università di Roma
Roma, Italy

[2]Yahoo! Research Labs
Barcelona, Spain

ABSTRACT

Defining a measure of similarity between queries is an interesting and difficult problem. A reliable query-similarity measure can be used in a variety of applications such as query recommendation, query expansion, and advertising.

In this paper, we exploit the information present in query logs in order to develop a measure of semantic similarity between queries. Our approach relies on the concept of the query-flow graph, a graph-based representation of a query log. The query-flow graph aggregates query reformulations from many users: nodes in the graph represent queries, and two queries are connected if they are likely to appear as part of the same search goal. Our query-similarity measure is obtained by projecting the graph (or appropriate subgraphs extracted from it) on a low-dimensional Euclidean space. Our experiments show that the measure we obtain captures a notion of semantic similarity between queries and it is useful for diversifying query recommendations.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

1. INTRODUCTION

Finding a measure of similarity between queries can be very useful to improve the services provided by a search engine. First, the ability to identify similar queries is in the core of any query-recommendation system. Second, query similarity can be used for expanding query expansion and enhancing search results. Additionally, a reliable notion of query-similarity can be used for broad matching of advertisements to queries, or even for suggesting keywords to advertisers. However, defining a query-similarity measure is not an easy task as it strongly depends on user intent: syntactically similar queries may originate from completely different intents. Issues such as polysemy, synonymy, high levels of noise, and the very small amount of available information make the problem challenging.

In such a complex setting, information extracted from query logs has shown to be effective. The information on how users interact with search engines has often been used to improve the user search experience. In particular, query-log analysis is used to obtain insights on how users refine their queries, and what kind of search strategies they are using to locate the information they need.

In this paper we describe a method of obtaining a query-similarity measure, based on query-log analysis. Our method relies on an aggregated representation of a query log by the means of a reformulation graph, which is known as the *query-flow graph* [6]. In this graph, nodes represent queries and two queries are connected if they are likely to appear as part of the same search *goal* [16].

Our main intuition is that related queries will tend to cluster in local neighborhoods of the query-flow graph. *Graph-projection methods* are known to map graph nodes into geometric spaces so that the distance distortion is minimized. Thus, we suggest *projecting* the query-flow graph (or appropriately defined subgraphs of it) and then measure query similarity on the resulting geometric space. The technique is general and it can be applied to other graphs obtained from query-logs, for example, the click graph [5, 11, 22].

We use the resulting query-similarity measure for a concrete application, namely for diversifying query recommendations. Query-recommendation systems are provided by all major search engines and they aim at helping users to find more easily what they are searching for. The idea is that a diversification algorithm requires a notion of query similarity, for which we use the proposed measure. We show that the resulting system provides diverse yet relevant recommendations.

Our main contributions are summarized as follows:

- we describe a method for measuring similarity between queries by projecting a query reformulation graph;
- we show that our similarity measure captures the human notion of related queries better than other measures on the original graph;

*Part of this work was done while visiting Yahoo! Research Labs, Barcelona

- we apply this method to the task of producing diverse and useful recommendations;
- we show how to improve its efficiency further, by projecting only the neighborhood of the input query.

Roadmap. The rest of this paper is organized as follows: Section 2 introduces related work about this topic, Sections 3 and 4 describe the framework we use to define our measures of query similarity, and Section 5 explores different variants for optimizing our measures. Section 6 describes our application for diversifying query recommendations and Section 7 offers our concluding remarks.

2. RELATED WORK

Query graphs. Graphs might be used to provide a navigable, compact representation of the query-related information extracted by query-logs. Query graphs of different types have been extensively studied in literature [2, 5, 3, 22, 6]. In most of these studies query-logs are represented by query-query graphs, where queries form the set of vertex and edges among queries capture various types of query-information. Baeza-Yates [2] articulates a first query-graph taxonomy, introducing five types of graphs where arcs connect queries according to different criteria (e.g., common words, common clicked URLs). Tiberi et al. [3] show how to infer semantic relations between queries from the cover graph and describe an application to the task of detecting multi-topical URLs.

Query-document graphs, also known as *click graphs* [5, 22] are bipartite graphs $G = \{Q, D, E\}$ where Q is the set of queries and D is the set of documents. A query is connected to documents that were clicked in the associated result list.

The query-flow graph [6] aggregates different sources of information in order to capture the latent query behavior of users. Each link is labeled with the probability that its endpoints are related. Leven and Loizou [20] introduce a concept similar to the query-flow graph, but their work is focused on browsing behavior inside a Web site. Borges and Levene propose an improved method for measuring the ability of a variable-length Markov model to summarize user Web navigation sessions up to a given length [8].

Query recommendations. Most query-recommendation methods use similarity measures obtained by mining (*i*) the query terms, (*ii*) the clicked documents, and/or (*iii*) the user sessions containing the queries. Typical methods use a combination of these factors.

Query recommendation based on clicked documents. Baeza-Yates et al. [4] find, given a query, related queries issued by other users and build query expansion methods to construct artificial queries. Their technique is based on a term-weight vector representation of queries, obtained from the aggregation of the term-weight vectors of the URLs clicked after the query. Wen et al. [30] also present a clustering method for query recommendation that is centered around various notions of query distance.

Craswell and Szummer [11] describe a method based on random walks on the query-click graph [5], and they test it for an application of image search. Fuxman et al. [13] use the query-click graph to find related keywords for advertising. Antonellis et al. [1] also use the query-click graph, and exploit the idea of co-citation through its generalization known as SimRank [15]. Mei, Zhou and Church [22] use a computation of hitting time for ranking related queries.

Query recommendation based on query reformulations. Many effective approaches focus on the analysis of user query sessions [12, 33, 17]. Fonseca et al. [12] propose a query recommendation system based on association rules applied to query logs. Zhang and Nasraoui [33] represent each user session by a complete graph where consecutive queries are connected with an edge of a predefined weight d. Not consecutive queries are connected by an edge weighted with the product of the weights on the walk connecting them. Recent works have shown that not only the previous query, but also the long-term interests of users, are important for understanding their information needs [21, 26].

In [17], the notion of query substitution is introduced: for each query, a set of similar queries is obtained by replacing the whole query or only its sub-phrases. White et al. [31, 32] use the query rewrites observed in a query log to generate query recommendations. Sadikov et al. [27] have recently proposed to cluster the refinements of a user query by performing a random walk on a query-document graph that incorporates both session and click information.

3. PRELIMINARIES

3.1 The query-flow graph

A query-flow graph, as defined by Boldi et al. [6], 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 plus two special nodes s and t, representing a *starting state* and a *terminal state* of any user search task;
- $E \subseteq V \times V$ 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 $w(q, q')$.

In the query-flow graph, two queries q and q' are connected by an edge if there is at least one session of the query log in which q' follows q. The weight w may depend on the application; in the following we simply consider the weight to be the frequency of the transition in the query log.

The edge probabilities along with other data associated to each transition, are used to segment *physical sessions* into *missions* [16] or *chains* [25]. Here, physical sessions are defined as sequences of the activities of a single user before a timeout of 30 minutes, while missions/chains are defined as sequences of activities that are topically related. This step is important for applications aimed at improving the user search experience.

3.2 Spectral projection

The query-flow graph captures implicit similarity between queries: queries connected by a heavy edge are similar in the sense that they are motivated by the same user information need. Query logs collected over a few months are rich in information, but they also contain a lot of noise. Our approach is motivated by the idea of defining a similarity measure between queries that takes into account the global structure of the query-flow graph, instead of taking into account only pairs of queries.

Measuring distances of nodes on large graphs is a well-studied problem, and many approaches have been proposed, including the *shortest-path* distance and the *commute time.* A drawback of those measures is that they are very expensive to compute, as their complexity is at least linear to the number of the nodes in the graph, while one would like to have measures whose complexity depends only on the nodes under consideration (and possibly features of the nodes) but not on the whole dataset.

One of the key methods for measuring distances in a graph, which has been used extensively for visualizing graph data, is the idea of *graph projection*. The idea is to project the original graph into a low-dimensional Euclidean space and then measure distances between graph nodes by considering the distances of the corresponding projected points. There are many techniques that can be used to obtain such projections, including multidimensional scaling [19], spectral projections [10, 18, 24], IsoMap [29], maximum-variance unfolding [28], and many more. In this paper we use the spectral projection, which we briefly describe below:

1. Given a graph G with adjacency matrix A, the Laplacian matrix $\mathcal{L}_G = D - A$ is computed using the diagonal matrix D, whose entry d_{ii} equal to the degree of the i-th node of G.
2. An embedding $\phi : V \to \mathbb{R}^m$ is computed by finding the $(m+1)$ eigenvectors of $\mathcal{L}_G$ that correspond to the smallest eigenvalues. Only m of these are used as the one corresponding to the smallest eigenvalue is the vector of all $1'$s.

This spectral embedding, known as *Fiedler embedding*, has the property of preserving the distances in the projected space. For "near-by" nodes u and v in the graph G, the Euclidean distance between the vectors $\phi(u)$ and $\phi(v)$ is small. Details on the properties of spectral embeddings of graphs and spectral algorithms can be found in [10, 18, 24].

For spectral projection, the notion of "near-by" nodes in the graph is related to the expected time of coming across node v in a random walk starting from node u. Hence, we expect that queries that are similar are projected to points that are relatively close in the embedding. Regarding the property of maintaining distances, the spectral projection optimizes a global objective function that can be interpreted as the overall distortion of the graph projection.

On the projected space, various distance metrics can be used, for example the Euclidean distance or the cosine distance. In our experiments, we observed that cosine similarity outperforms Euclidean distance, hence, in the rest of the paper we focus on cosine similarity. Note, that since the projected vectors may contain negative numbers, the cosine between two vectors can be negative. In order to obtain a measure in $[0, 1]$ we rescale the cosine as follows:

$$\mathrm{Sim}(q, q') = \frac{1 + \cos(q, q')}{2}.$$

Recently the notion of directed Laplacian was introduced and analyzed by Fan Chung [9]. Since the query-flow graph is a directed graph, using the directed Laplacian is more appropriate for our application scenarios. However, our evaluation showed that the projection based on the directed Laplacian does not yield any improvement.

In the next two sections we describe our experiments with projections of the query-flow graph. Our objective is to explore systematically the space of possible parameters and alternatives on defining appropriate subgraphs to project, in order to optimize the quality of query similarities.

We note that we have not tried to explore the possibility of improving our empirical results using different graph embedding algorithms. Any graph embedding algorithm can be used as a black-box in our method, instead, our main focus has been to leverage the idea that graph projections can yield meaningful notions of query similarity, and we have also experimented extensively with finding the best subgraphs to project.

Table 1: Example of manually-built clusters

Query	Clusters
sun	**1.** the sun newspaper, mirror, times
	2. earth, mars, mercury
	3. sun java, sun microsystems
stone	**1.** stone weight, stone measurement
	2. rock, granite
	3. stone brick, stone masonry
spoiler	**1.** movie spoiler, tv show spoiler
	2. car spoiler, custom car spoiler

4. FRAMEWORK FOR STUDYING QUERY SIMILARITY

4.1 Dataset

The query-flow graph was built using a set of sessions extracted from a query log from the Yahoo! search engine.

We improve the graph by estimating the probability that both the queries q and q' in a transition belong to the same "search mission" [16] (also known as "query chain" [25]). This modification prunes out transitions to frequent navigational queries such as popular web portals.

Motivated by the results of some preliminary assessments, we apply a number of filters on the graph, such as removing the transitions that have frequency less than 5. We also remove s and t and prune all the nodes that remain isolated: these correspond to sessions composed by singleton queries. Altogether, starting from a graph with 58 312 610 nodes and 131 836 560 edges, we obtain a graph with 4 152 773 nodes and 7 788 232 edges. This is the graph G that we consider in the rest of the paper.

4.2 Evaluation method

Our evaluation method is summarized as follows: We select 140 queries from the log, sampling queries that are neither too frequent nor too infrequent (*torso queries*), as head queries tend to be of a particular type (e.g., navigational), while tail queries give information that is too sparse. We focus on single-term queries that are likely to have more than one interpretation. For each of these queries, we build a *test set* of related queries by selecting a small set of their most frequent successors in the query-flow graph. Each test set is then clustered into 2 to 5 clusters by human editors; the editors decided the number of clusters for each set.[1] The clustering of a test set represents the ground truth for that set. Table 1 shows examples of clustered test sets. We then apply our graph projection method, and we obtain a similarity measure for queries, for which we evaluate its agreement with the human-defined clustering. We compare different

[1]The clustering of the query test sets were done by four editors. We tested whether our results depend on the subjective perception of the editors about clustering, by repeating our analysis for the queries labeled by each editor separately. We found that in all cases we obtain similar results. We omit the details for lack of space.

variants of graph projections by testing which variants yields similarities that agree better with the golden truth.

Note that to apply our methodology we need to measure agreement between a clustering and a similarity function. We use the following measure: let V be a set and $\mathcal{C} = \{C_1, \ldots, C_k\}$, with $V = \bigcup_i C_i$, be a clustering of V. For a similarity function $\mathrm{Sim}(q, q')$, we introduce two scores:

intra-cluster similarity of cluster C_i:

$$\mathrm{InSim}(C_i) = \sum_{q_h, q_j \in C_i, h \neq j} \frac{2 \cdot \mathrm{Sim}(q_h, q_j)}{|C_i||C_i - 1|}$$

inter-cluster similarity of cluster C_i:

$$\mathrm{OutSim}(C_i) = \sum_{l=1\ldots k, l \neq i} \left[\sum_{q_h \in C_i} \sum_{q_j \in C_l} \frac{2 \cdot \mathrm{Sim}(q_h, q_j)}{|C_i||C_l|} \right]$$

The intuition is that a similarity measure agrees with clustering $\mathcal{C}$ if the InSim score is large compared to the OutSim score. Thus, we capture the quality of a similarity measure, with respect to the clustering $\mathcal{C}$, using the ratio

$$\mathcal{M}_{\hat{\mathcal{C}}}(\mathrm{Sim}) = \frac{\mathbf{E}[\mathrm{InSim}(C)]_{C \in \hat{\mathcal{C}}}}{\mathbf{E}[\mathrm{OutSim}(C)]_{C \in \hat{\mathcal{C}}}}.$$

Given two different similarity measures, the best one is the one that maximizes the measure $\mathcal{M}_{\hat{\mathcal{C}}}(\mathrm{Sim})$.

4.3 Sub-graph construction method

We experiment with two alternatives for extracting subgraphs from the query-flow graph: query-dependent and query-independent method.

Query-dependent subgraphs. Given a query q we extract a subgraph around q by a breadth-first search from q. For a graph $G = (V, E)$ and two nodes q, q', let $d(q, q')$ be the length of the shortest path from q to q' following directed edges in E. Let $V_d(q) = \{q' \in V : d(q, q') \leq d \vee d(q', q) \leq d\}$. For instance, $V_0(q) = \{q\}$ and $V_1(q)$ contains the in-neighbors and out-neighbors of q. We define the sets:

- $E_d(q) = \{(q_1, q_2) \in E : q_1 \in V_d(q) \wedge q_2 \in V_d(q)\}$
- $O_d(q) = \{(q_1, q_2) \in E : q_1 \in V_{d-1}(q) \wedge q_2 \in V_d(q)\}$
- $I_d(q) = \{(q_1, q_2) \in E : q_1 \in V_d(q) \wedge q_2 \in V_{d-1}(q)\}$.

We experimented with the following subgraphs of q:

- $F_d(q) = (V_d(q), E_d(q))$
- $S_d(q) = (V_d(q), E_{d-1}(q) \cup O_d(q) \cup I_d(q))$

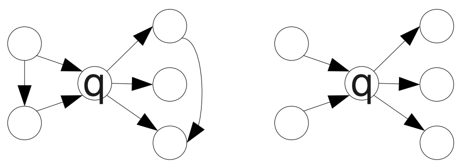

Figure 1: Example $F_1(q)$ and $S_1(q)$.

Figure 1 shows an example. The sizes of the obtained subgraphs vary widely, with $|V_2(d)|$ ranging from 31 to 20 702 queries (median: 2 320 queries).

Query-independent subgraphs by partitioning. Query-dependent subgraphs may be expensive to compute at query time, so we also experimented with query-independent subgraphs obtained by partitioning the filtered graph. The partitioning was done using METIS [23] and varying the number of clusters that METIS takes as a parameter.

5. OPTIMIZING THE SIMILARITY MEASURE

The objective of our study is twofold: (i) to demonstrate the effectiveness of the query-flow graph projections in order to define a measure able to capture the human notion of similarity between queries, and (ii) to optimize such a query-similarity measure.

To address the first objective we define a baseline measure that relies on the query-flow graph but does not use projections. To obtain more refined graphs, and thus similarities of better quality, we apply the projection method locally, on the neighborhood of a given query. This approach is based on constructing query-dependent subgraphs as we discuss in detail in Section 4. Applying projections locally yield a better similarity measure, but unfortunately, the method is computationally more expensive since it requires to build a different subgraph for each query. Thus we propose a "hybrid" approach, which builds query-independent subgraphs by using partitions of the query-flow graph obtained with the METIS graph-clustering algorithm.

Overall, we perform a large number of experiments to assess the following parameters: (i) the number of dimensions of the spectral embedding, (ii) the choice of the weighting scheme, (iii) the method to be used in the construction of the query-dependent subgraphs, and (iv) the number of partitions to be considered in the computation of query-independent subgraphs.

5.1 Similarity without graph projections

Our first similarity measure is purely based on the query-flow graph: the similarity between two queries is given by the cosine similarity of their vectors of neighbors. Given a query q, we denote by $N(q)$ its vector of neighbors in the query-flow graph.

For our test sets, the average value of $\mathcal{M}_{\hat{\mathcal{C}}}$ for this baseline similarity measure is 1.03 with a variance of 0.03. Figure 4(a) reports the results obtained for the query *watch*; we observe little separation between queries in different clusters. These results suggest that the metric based on the cosine similarity of the neighborhoods of two queries does not capture well the similarity of queries. We stress here the fact that methods based on random walks [11] or hitting time [22] are not able to capture the notion of semantic similarity we aim to. In the example of Figure 4(a), the probability to end up in frequent queries like "`rolex watch`" or "`watch free movies online`", starting from the query "`watch`", might be quite similar even if such queries are not semantically related.

5.2 Similarity with graph projections

A quite natural objection to the use of the Fiedler embedding of the query-flow graph is that this projection is intended to be applied on undirected graphs. Surprisingly preliminary experiments, which we do not present for lack of space, showed that using the directed Laplacian does not yield any improvement. Hence, we retained the standard method. We assume that similar queries are projected onto points that are close in the embedded Euclidean space. We then measure the distance between two queries in terms of cosine distance between the corresponding vectors.

The extensive set of experiments we performed confirms that the distance measure defined by means of the spectral projection of the query-flow graph into a lower-dimensional

Table 2: Average $\mathcal{M}_{\hat{C}}$ for different projections with different number of dimensions

	Dimensions			p-value
Method	3	5	7	(3 vs 7)
S_2	1.9 ± 1.5	1.8 ± 1.2	1.3 ± 0.8	0.14
S_3	3.3 ± 6.5	2.2 ± 2.5	1.8 ± 1.3	0.12
G	3.2 ± 9.1	1.3 ± 0.9	1.3 ± 0.9	0.19

Table 3: Average $\mathcal{M}_{\hat{C}}$ obtained for $S_2(q)$, $S_3(q)$ using different weighting schemes.

	Weighting			Significant differences	
	bin.	log	raw	0.1 * 0.05 ** 0.01 ***	
S_2	1.9	1.8	1.3	bin.>raw ***	log>raw ***
S_3	2.2	1.9	1.7	bin.>raw **	bin.>log *

space captures the notion of similarity among queries better than the one relying on the original graph.

Choice of dimensions. In order to investigate how varying the number of dimensions of the spectral projection affects performance, we build three embeddings of G, which respectively have $m = 3$, $m = 5$ and $m = 7$ dimensions. To get a more complete picture, we do the same for two types of query dependent subgraphs, S_2 and S_3. For each of these cases, we use the projections to compute the similarity between all the pairs of queries in every test set collected, and then we compute the measure $\mathcal{M}_{\hat{C}}$ to evaluate agreement with the clusters created by hand. We also perform t-tests to figure out whether the differences among the various cases are statistically significant or not.

Our results, reported in Table 2, show that increasing the number of dimensions does not determine a considerable gain in terms of quality. A slight improvement in terms of less variance is observed, but the differences between the various cases are not statistically significant. For this reason we fix $m = 5$ in the remainder of our experimentation.

Choice of weighting scheme. In the query-flow graph, every edge connecting two queries is weighted with the frequency of the transition in the original query log, and the transition probabilities of the edges can be used to identify queries that represent similar information needs. We perform a number of tests to investigate whether taking the edge weights into account during the computation of the spectral embeddings improves our method.

We experiment with three different weighting functions:

- **binary**: $w : E \to \{0, 1\}$ s.t. $w(q, q') = 1$. This case corresponds to the baseline (unweighted graph).
- **raw-count**: $w : E \to \mathbb{N}$ s.t. $w(q, q') = c(q, q')$ ($c(q, q')$ is the occurrence count of the transition in the log).
- **log(count)**: $w : E \to \mathbb{R}$ s.t. $w(q, q') = \log(c(q, q'))$.

When an edge between two queries exists in both directions, we symmetrize the weights choosing the maximum of the two. This step is needed for the computation of the projection. We also tried with minimum, average or sum of the two weights, but we observed no substantial changes.

We compare the above weighting schemes for two types of query dependent subgraphs: $S_2(q)$ and $S_3(q)$. Table 3 shows the results. Binary and log weights yield the best results, whereas the usage of raw counts hurts performance, and the difference between this weighting scheme and the other two is statistically significant. For this reason we discard the raw counts, and we focus on the simplest of the two schemes that exhibit the best behavior, which is the one using binary weights. We consider projections of unweighted graphs in the remainder of this section.

Table 4: $\mathcal{M}_{\hat{C}}$ for query-dependent subgraphs.

	Average	Significant differences	
Method	$\pm$ st.dev	0.1 * 0.05 ** 0.01 ***	
N	1.02 ± 0.17		
G	1.22 ± 0.74	$> N$ ***	
F_1	1.49 ± 2.07	$> N$ **	
S_2	1.63 ± 0.88	$> N$ ***	$> G$ ***
S_3	1.68 ± 1.45	$> N$ ***	$> G$ ***

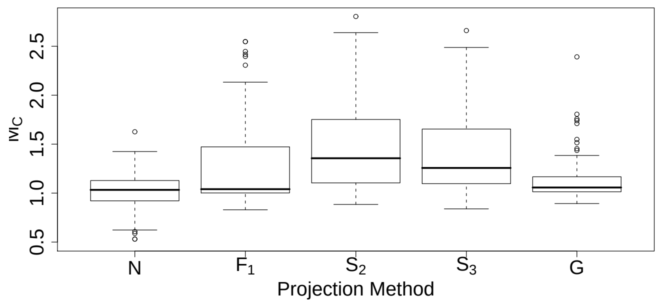

Figure 2: Comparison among local projections, projection of the full graph and direct use of the query-flow graph

5.3 Query-dependent subgraphs

We now study if projecting the neighborhood of a given query improves performance. We consider three types of query-dependent subgraphs: $F_1(q)$, $S_2(q)$, $S_3(q)$. The first is the subgraph induced by the neighbors of the input query, while the other two graphs are obtained by applying two or three steps of a breadth-first search on the query-flow graph, starting from the input query.

We compare the embeddings of the query-dependent subgraphs against the methods discussed before, i.e., projecting the whole graph or using the query-flow graph directly. The results of this study are presented in Figure 2 and Table 4: N is the baseline, while the other cases represent the methods that project either the full graph G or the corresponding query-dependent subgraphs. Local projections introduce a relative improvement in the average $\mathcal{M}_{\hat{C}}$ in the range of the $16\%-39\%$. Even if the variances of local projections are also increasing we can observe that the lower bound for both S_2 and S_3 match the upper bound of N. Also, the differences between the query-dependent subgraphs and the other two approaches are statistically significant (see Table 4).

Figure 5 reports a qualitative comparison of the above methods for the query *watch*.

5.4 Query-independent sub-graphs

Using local projections of small query-dependent subgraphs allows to define a metric that captures better the similarity between queries. However, this method has the clear drawback of requiring query-time computational processing, which may be expensive. For this reason we investigate

Table 5: $\mathcal{M}_{\hat{\mathcal{C}}}$ for varying numbers of clusters; none of the pair-wise differences is statistically significant

Method	Expansion distance	Number of clusters	Average $\pm$ st. dev
Full graph	-	-	1.2 ± 0.2
Clustering	1	100	1.0 ± 0.1
Clustering	2	100	1.2 ± 0.5
Clustering	2	200	2.0 ± 10.3
Clustering	2	1 000	4.2 ± 25.8
Clustering	2	5 000	3.9 ± 27.0
Clustering	2	20 000	6.3 ± 46.7

whether projecting larger, query-independent subgraphs allows to trade-off performance and computational costs.

We generate sets of query-independent subgraphs using the METIS algorithm, which partitions the nodes of a graph into balanced clusters minimizing the number of edges in the cut. The number of clusters to be created is chosen by the user. In the following, we briefly describe how we use a partition of the nodes in the query-flow graph created with METIS to derive query-independent subgraphs. We then compute the spectral embeddings of these subgraphs.

Cluster expansion. We first partition the graph into 100 and 200 clusters. We choose these values because they yield cluster sizes comparable to the size of query-dependent subgraphs for which our method obtains the best performance.

As first attempt, we directly project the partition created by METIS. This solution performs very poorly (results are omitted due to lack of space), because the raw clusters do not typically include a significant fraction of the neighborhood of each node. To overcome this limitation, we study how to make the partitions include (a significant fraction of) the neighborhood of each node assigned to them. We test two methods. The first strategy consists of enlarging each partition with the in/out-neighbors of every node originally included in it – so that clusters may overlap. We found that this method does not improve performance.

We then experiment with a more expensive strategy, which consists of adding to each partition the two-step neighborhood of every node originally assigned to it. This solution creates larger clusters, and experimental evaluation shows that it provides results comparable to those obtained by projecting the full graph (see Table 5). Hence, we retain this approach as our cluster expansion method.

Choice of number of clusters. The expansion step is necessary to make the clusters include many neighbors of a given node, i.e., queries that are likely to be related to the input query. However, this operation creates clusters of very large size: we believe that this can be a reason for not having a significant improvement in performance. Hence, we perform more extensive experiments, using METIS to divide the query-flow graph into a larger number of partitions and applying the expansion step to the sets of nodes obtained. Table 5 shows how the method behaves with 1 000, 5 000 and 20 000 clusters. Starting from 5 000 we get a little improvement in performance, but the differences are not significant at $p < 0.1$. Although the method performs at least as well as the projection of the full graph, our intuition is that it would be worth exploring other approaches to extract query-independent subgraphs from the query-flow graph.

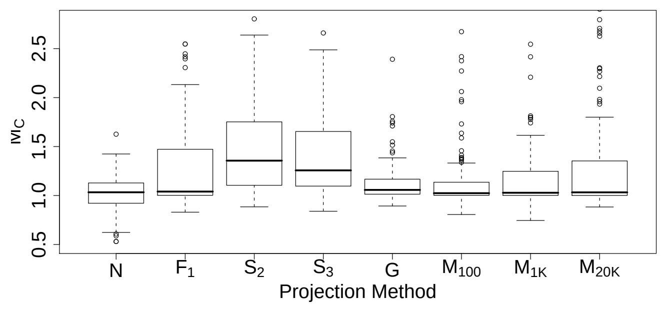

Figure 3: Summary of performance of some systems

5.5 Summary

Figure 3 summarizes the tested methods, which can be divided into three groups:

- Using query-flow graph without projection (N): this performs close to random;
- Projecting the whole graph (G) or query independent clusters (M_K): good performance, with a possible advantage to systems based on clustering;
- Projecting query-dependent sub-graphs (F_d or S_d): performs the best, with a small advantage for the systems that expand the neighborhood at 2 or 3 steps over the subgraph induced by the direct neighbors.

With respect to effectiveness, a positive result is that for all the projection-based methods we tested, in at least 75% of the cases the system gave a larger similarity to queries in the same user-defined clusters than to queries in different user-defined clusters.

6. APPLICATION TO QUERY RECOMMENDATIONS

In this section we describe how we apply the proposed method for producing diverse query recommendations. Diversification of search results or query recommendations is a strategy adopted by search engines to improve the user experience and minimize the risk that the information need of the user will not be satisfied.

For our experiment we use a random sample of 100 queries. For each query, we generated a set of baseline query recommendations using a method suggested in [7]. This method (QUERYFLOW-SP) associates a query q with a set $\mathcal{Q}$ of recommendations obtained by selecting the most frequent reformulations from q. Each query $q' \in \mathcal{Q}$ is assigned a ranking score given by the frequency of the transition (q, q'). This method was shown to perform as well as more sophisticated recommendation algorithms. In the following, we refer to it as the baseline.

Next, we use the diversification method described by Gionis et al. [14]. The idea is to apply a greedy search that maximizes diversity while maintaining high relevance. The algorithm takes as input the set $\mathcal{Q}$ and builds a diverse set $\mathcal{A}$ of queries. First, the query $q_0 \in \mathcal{Q}$ with the highest relevance score is selected, removed from $\mathcal{Q}$, and inserted into $\mathcal{A}$. Observe that this ensures that the most popular query related to the input query is always selected for recommendation. Next, the algorithm starts an iterative phase: at each step the query $q \in \mathcal{Q}$ with maximum score $s(q)$ is removed from

Query: watch	rolex watch	citizen watch	seiko watch	watch free movie online	watch movies.net
rolex watch	-	0.052	0.044	0.003	0.002
citizen watch	0.052	-	0.087	0.004	0.002
seiko watch	0.044	0.087	-	0.004	0.003
watch free movie online	0.003	0.004	0.004	-	0.04
watch movies.net	0.002	0.002	0.003	0.04	-

(a) Method: N, i.e., cosine similarity between vectors of neighbors in the QFG

Query: watch	rolex watch	citizen watch	seiko watch	watch free movie online	watch movies.net
rolex watch	-	0.99	0.99	0.12	0.12
citizen watch	0.99	-	1.00	0.09	0.08
seiko watch	0.99	1.00	-	0.12	0.11
watch free movie online	0.12	0.09	0.12	-	1.00
watch movies.net	0.12	0.08	0.11	1.00	-

(b) Method: spectral embedding of G

Figure 4: Example: query similarities using cosine similarity of neighbors (no projection) and projection of the full graph for the query *watch* . Lines separate manually-assigned clusters for these queries.

Query: watch	rolex watch	citizen watch	seiko watch	watch free movie online	watch movies.net
rolex watch	-	1.00	1.00	-0.57	-0.58
citizen watch	1.00	-	1.00	-0.56	-0.57
seiko watch	1.00	1.00	-	-0.57	-0.57
watch free movie online	-0.57	-0.56	-0.57	-	1.00
watch movies.net	-0.58	-0.57	-0.57	1.00	-

Figure 5: Example query similarities using projection of the subgraph $S_2(q)$ for the query *watch*

$\mathcal{Q}$ and inserted into $\mathcal{A}$. The score $s(q)$ is a combination function of the relevance of q with the distance $d(q, \mathcal{A})$ of query q from the set $\mathcal{A}$ of queries that have already been selected. The algorithm balances diversity with relevance. Given that the two measures, distance from other queries and relevance, are not comparable, the algorithm tries to maximize the product of the two, picking up queries that have a high ranking score while being not too similar to the queries that have already been selected. We omit the details of the diversity algorithm since it is not the focus of this paper. We derive the distance metric that we use for diversification from our projection method. We experiment with three schemes: N, G and $S_2(q)$. These methods measure similarity between queries in terms of cosine similarity between (a) their vectors of neighbors in the query-flow graph; (b) the vectors associated with the queries in the projection of the full graph; (c) the vectors obtained projecting the subgraph $S_2(q)$. In the case of N, we define the distance of a query q from the set $\mathcal{A}$ as the minimum distance between q and a query in $\mathcal{A}$:

$$d(q, \mathcal{A}) = min_{t \in A}\{d(\mathbf{v}(q), \mathbf{v}(t))\}.$$

In the case of G and $S_2(q)$ the distance of q from $\mathcal{A}$ is defined as the distance between q and the centroid $\mathbf{c}(\mathcal{A})$ of the set $\mathcal{A}$.

$$d(q, \mathcal{A}) = d(\mathbf{v}(q), \mathbf{c}(\mathcal{A})).$$

Perceived diversity. The task is highly subjective, and when measuring the agreement of the assessors on a subset of questions in which they overlap, we observe a moderate level of agreement (Cohen's $\kappa = 0.49$).

In Table 6 we show the results of this evaluation. The results are expected given the results from previous sections, as $S_2(q)$ is the best method (in 51% of the cases it is perceived as more diverse than the baseline, in 14% of the cases as less diverse), followed by G (projecting the full graph), followed by N (using the graph without applying projection).

Table 6: Result of user test for assessing diversity of recommendations

	Prob. B is more/less diverse than A		
	N	G	S_2
Baseline	0.30/0.13 $*$	0.39/0.25 $*$	0.51/0.14 $***$
N	-	0.42/0.25 $**$	0.49/0.15 $***$
G	-	-	0.46/0.25 $**$

Significance: 0.1 $*$ 0.05 $**$ 0.01 $***$

Perceived relevance. We examine 100 queries and take the union of the top-3 recommendations from all the systems that are compared. This yields 460 distinct query pairs. The assessment is to measure if the recommendation is *relevant* to the original query. In this case, the inter-assessor agreement is $\kappa = 0.53$.

The accuracy of relevant queries that the baseline algorithm recommends is 97%. When using the projection on the full graph, this figure drops to 90%, and to 92% when using the S_2 method. Instead, when using N there was basically no drop in relevance, measuring a 97% of recommendations relevant to the original query. These results suggest that (i) N does only a small change in the recommendations, and (ii) S_2 and the method that projects the full graph change the recommendations but still keep the fraction of recommended queries that are relevant to the original query at 90% or more.

7. CONCLUSIONS

We have shown that projecting the reformulation graphs in a low-dimensional space allowed us to define a similarity measure between queries. To the best of our knowledge, this is the first attempt to apply spectral methods to query-reformulation analysis.

After methodically exploring several design choices, we found two methods that perform well: one for off-line processing and one for on-line processing. The method for off-line processing basically works by storing 3-5 spectral coordinates per query and then using them at query time at constant cost. The method for on-line processing requires computing a small subgraph at query time and then projecting this graph to obtain the coordinates. Our experiments suggest that the on-line method is a more effective similarity measure, but of course it has a higher computational cost.

To demonstrate the practical impact of our method, we tested our measure as a component of a system for producing diverse query recommendations. Our experiments show that our method can be used to produce diverse recommendations at small cost of relevance.

As future work, we would like to seek more effective off-line methods than the projection on the full graph. We also plan to investigate alternative projection methods, as well as the use of other query graphs, such as the click graph.

Key references: [6, 10].

8. REFERENCES

[1] Antonellis, I., Garcia-Molina, H., and Chang, C.-C. Simrank++: Query rewriting through link analysis of the click graph. In *VLDB* (2008).

[2] Baeza-Yates, R. Graphs from search engine queries. In *Theory and Practice of Computer Science (SOFSEM)* (2007).

[3] Baeza-Yates, R., and Tiberi, A. Extracting semantic relations from query logs. In *KDD* (2007).

[4] Baeza-Yates, R. A., Hurtado, C. A., and Mendoza, M. Query recommendation using query logs in search engines. In *EDBT Workshops* (2004).

[5] Beeferman, D., and Berger, A. Agglomerative clustering of a search engine query log. In *KDD* (2000).

[6] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., and Vigna, S. The query-flow graph: Model and applications. In *CIKM* (2008).

[7] Boldi, P., Bonchi, F., Castillo, C., Donato, D., and Vigna, S. Query suggestions using query-flow graphs. In *WSCD* (2009).

[8] Borges, J., and Levene, M. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Trans. Knowl. Data Eng. 19*, 4 (2007), 441–452.

[9] Chung, F. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics 9*, 1 (2005).

[10] Chung, F. R. K. *Spectral Graph Theory (CBMS Regional conf. Series in Mathematics, No. 92)*. American Mathematical Society, February 1997.

[11] Craswell, N., and Szummer, M. Random walks on the click graph. In *SIGIR* (2007).

[12] Fonseca, B. M., Golgher, P. B., de Moura, E. S., and Ziviani, N. Using association rules to discover search engines related queries. In *LA-WEB* (Washington, DC, USA, 2003).

[13] Fuxman, A., Tsaparas, P., Achan, K., and Agrawal, R. Using the wisdom of the crowds for keyword generation. In *WWW* (2008).

[14] Gionis, A., Manca, M., Pintus, E., Castillo, C., and Donato, D. Diversity in web search, Technical Report, Yahoo! Research.

[15] Jeh, G., and Widom, J. Simrank: a measure of structural-context similarity. In *KDD* (2002).

[16] Jones, R., and Klinkner, K. L. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM* (2008).

[17] Jones, R., Rey, B., Madani, O., and Greiner, W. Generating query substitutions. In *WWW* (2006).

[18] Koren, Y. On spectral graph drawing. In *COCOON* (2003).

[19] Kruskal, J. Nonmetric multidimensional scaling: A numerical method. *Psychometrika 29*, 2 (1964).

[20] Levene, M., and Loizou, G. A probabilistic approach to navigation in hypertext. *Inf. Sci. 114*, 1-4 (1999), 165–186.

[21] Luxenburger, J., Elbassuoni, S., and Weikum, G. Matching task profiles and user needs in personalized web search. In *CIKM* (2008).

[22] Mei, Q., Zhou, D., and Church, K. Query suggestion using hitting time. In *CIKM* (2008).

[23] Metis - Family of multilevel partitioning algorithms. http://glaros.dtc.umn.edu/gkhome/views/metis/.

[24] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *NIPS* (2001).

[25] Radlinski, F., and Joachims, T. Query chains: learning to rank from implicit feedback. In *KDD* (2005).

[26] Richardson, M. Learning about the world through long-term query logs. *ACM Trans. Web 2*, 4 (2008).

[27] Sadikov, E., Madhavan, J., Wang, L., and Halevy, A. Clustering query refinements by user intent. In *19th International World Wide Web Conference, WWW 2010.* (2010).

[28] Sun, J., Boyd, S., Xiao, L., and Diaconis, P. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review 48* (2004), 2006.

[29] Tenenbaum, J. B., Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000).

[30] Wen, J.-R., Nie, J.-Y., and Zhang, H.-J. Clustering user queries of a search engine. In *Proc. of the 10th WWW conf.* (2001).

[31] White, R. W., Bilenko, M., and Cucerzan, S. Studying the use of popular destinations to enhance web search interaction. In *SIGIR* (2007).

[32] White, R. W., Bilenko, M., and Cucerzan, S. Leveraging popular destinations to enhance web search interaction. *ACM Trans. Web 2*, 3 (2008), 1–30.

[33] Zhang, Z., and Nasraoui, O. Mining search engine query logs for query recommendation. In *Proc. of the 15th WWW conf.* (2006).