From Machu_Picchu to *"rafting the urubamba river"*: Anticipating information needs via the Entity-Query Graph

Ilaria Bordino[1] Gianmarco De Francisci Morales[2] Ingmar Weber[3*] Francesco Bonchi[4]

[1,2,4]Yahoo! Research Barcelona, Spain [3]Qatar Computing Research Institute, Doha, Qatar

{[1]bordino, [2]gdfm, [4]bonchi}@yahoo-inc.com [3]ingmarweber@acm.org

ABSTRACT

We study the problem of anticipating user search needs, based on their browsing activity. Given the current web page p that a user is visiting we want to recommend a small and diverse set of search queries that are relevant to the content of p, but also non-obvious and serendipitous.

We introduce a novel method that is based on the content of the page visited, rather than on past browsing patterns as in previous literature. Our content-based approach can be used even for previously unseen pages.

We represent the topics of a page by the set of Wikipedia entities extracted from it. To obtain useful query suggestions for these entities, we exploit a novel graph model that we call EQGraph (*Entity-Query Graph*), containing entities, queries, and transitions between entities, between queries, as well as from entities to queries. We perform Personalized PageRank computation on such a graph to expand the set of entities extracted from a page into a richer set of entities, and to associate these entities with relevant query suggestions. We develop an efficient implementation to deal with large graph instances and suggest queries from a large and diverse pool.

We perform a user study that shows that our method produces relevant and interesting recommendations, and outperforms an alternative method based on reverse IR.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.3 **Information Search and Retrieval.**

Keywords

Query Suggestions, Implicit Search, Serendipity, Entity Extraction.

1. INTRODUCTION

Exploring the Web is often a *serendipitous* experience, where users with no pre-existing search objective meander from topic to topic, sometimes discovering unexpected, surprising, and interesting information [16]. Such serendipitous search encounters can be extremely enriching for the users, as they may stimulate their thinking to arrive to particular creative insights [2, 32]. In this scenario, it is interesting to analyze the class of web search queries that are *triggered* by the content of a previously read web page. For example, while reading a news article about the 2012 presidential election results in France, the user might want to read more about the new president and his financial policies, thus issuing a query such as "francois hollande financial policies". Figure 1 shows a real example from our experiments.

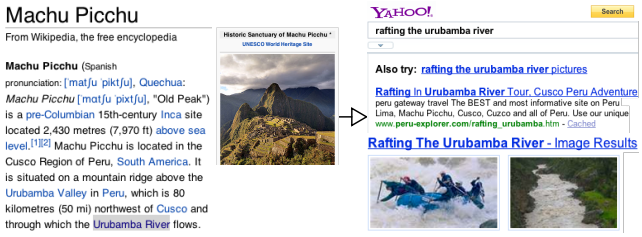


Figure 1: Example of a suggestion produced by our method in our experiments: when reading the Wikipedia page about Machu Picchu, a pre-Columbian Inca site in Peru, we suggest to the user a query about rafting the Urubamba, the river that flows in the valley where Machu Picchu is located.

In this paper we tackle the problem of *anticipating* such newly created information needs. More concretely, given the current web page p that a user is visiting we want to recommend a small and diverse set of search queries that are relevant to the content of p, non-obvious and serendipitous.

Observe that this problem is not just a web search problem *in reverse*, as we are not trying to identify queries that would lead to the page p being ranked highly in the result list. Such queries are mostly obvious given the content of the page, and this is not desirable for our application. Our goal is to identify queries that *stem from* the page, rather than queries that *lead to* the page.

The problem of understanding users' intent and supporting them in the formulation of web search queries has been studied extensively over the last years [3, 4, 6, 13, 30]. However, the standard problem studied is that of recommending new related queries that refine a given search query. In other terms, in the traditional context it is assumed that the user has already explicitly given an indication concerning her intent through a (possibly incomplete) web search query. In this perspective the provided recommendations are *passive*,

*This work was done while the author was at Yahoo! Research Barcelona.

WSDM'13, February 4–8, 2013, Rome, Italy.

in the sense that they only come after a user has submitted one or more queries to a search engine. The reasons behind these queries and what triggered the information need are not taken into account.

Instead, in the scenario we consider in this paper, the user does not need to formulate a query that represents explicitly an information need. The idea is that the user is picked up at an earlier stage during the search process, when she is reading a web page and, potentially, she has not even yet conceived the thought of issuing a web search query at all. The motivation for considering such a scenario is that many information needs of web users are actually triggered by what they browse. This particularly holds for informative pages, such as news and blogs: whenever a user finds something interesting or unclear while reading a page, she might want to go deeper in the matter and, to obtain further information, she might formulate a query.

A recent work by Cheng et al. [10] has proposed to exploit data gathered from search logs and browsing logs to actively predict users' query intent based on the web pages they browse. The model proposed by Cheng et al. is based on the frequent patterns of (page, query) transitions. The basic idea is that of exploiting *"the wisdom of the crowd"*: if after visiting a particular page, many users continue with the same query then the original page is likely to "trigger" the query. One main limitation of this approach is that, being based on frequent patterns, it is more effective for popular pages that have been seen many times by many different users, while it is inherently inapplicable to previously unseen pages. Unfortunately, as discussed above, the kind of pages that are more likely to be an interesting playground for these recommendations are informative pages which are very dynamical in nature and usually have a short life: from few hours to few days. Therefore it is important to develop methods for recommending queries even for new pages.

In this paper we tackle such a goal, by focusing on the content of pages. In particular, given any web page, we build a succinct representation of its subject matter by extracting the main concepts described or discussed in it, in the form of Wikipedia entities. With the aim of obtaining relevant and interesting query recommendations for the entities contained in a page, we devise a novel graph model, which we call EQGraph (*Entity-Query Graph*).

Our model extends the well-known Query-Flow graph [5]. It consists of two distinct sets of nodes, namely Wikipedia entity nodes and query nodes, and three different sets of arcs: entity-entity, query-query, and entity-query. The query-query connections are akin to their counterpart in the Query-Flow graph: the presence of a directed arc between two query nodes indicates a sufficiently high likelihood of submitting the second query after having issued the first one. In the EQGraph each entity is connected to all the queries that contain it. Finally, the entity-entity transitions are equivalent to query-flow transitions at the conceptual level of entities. We draw a directed arc between two entities if a user who issued a query related to the first entity is likely to search for the second entity. We derive these entity-entity transitions from the query-flow transitions by aggregating transitions that are related to the same entities.

We leverage the EQGraph to recommend interesting queries for web pages. We start from the *seed set* of entities contained in the page. To ensure that the seed set contains a sufficiently large number of concepts, we expand the initial set of entities by performing Personalized PageRank computations in the subgraph of the EQGraph induced by all the entity nodes.

In the EQGraph, the entities of interest for a page are connected to the queries that contain them. However, we do not build our final query recommendations for the page by simply looking at these queries. Instead we perform a Personalized PageRank computation for the expanded set of entities in the full EQGraph.

We could use a textual representation of the entities to provide synthetically generated queries. However, we decide to employ real queries to leverage the wisdom of the crowd. Queries that are pointed by multiple entities are more likely to be interesting because they are at the intersection of different topics in a page. The EQGraph naturally captures this intuition: if two entities are related, there will likely be a query that is pointed by both entities.

The EQGraph can be huge. For our experiments, we have built an instance of the graph containing several hundreds of millions nodes and arcs. Running Personalized PageRank computations on such a large graph is potentially a serious performance bottleneck of our query recommendation method. To perform these computations in an efficient and scalable manner, we develop a Giraph[1] implementation of the Personalized PageRank algorithm. Giraph is a Hadoop-based framework for running graph algorithms on large-scale data in parallel by distributing the load on a large number of computers. It is based on the Bulk Synchronous Parallel (BSP) model of computation [34].

We test our method on a large EQGraph extracted from a recent (2012) sample of a Yahoo! query log . We produce query recommendations for a set of 150 pages sampled from Wikipedia, Yahoo! News and Yahoo! Finance. We evaluate the quality of the produced recommendations by performing a user study, and find that our method produces relevant and non-obvious suggestions.

The main contributions of this paper are the following:

- We introduce the EQGraph as a novel object. This object enriches the standard Query-Flow graph [5] with entity nodes. We describe this graph in detail as it has interesting properties in its own right and could be applied to other problems.
- We present a novel method for anticipating information needs that arise from the content of pages that people are browsing. Our method is based on Personalized PageRank computation over the EQGraph.
- We use Wikipedia entities to represent the information items contained in web pages. By leveraging entities, our model is able to generate query recommendations for previously unseen pages.
- We provide an efficient implementation that can deal with very large graphs. We develop a Giraph implementation of the Personalized PageRank algorithm, which could be useful for many other applications.
- We perform a user study that shows that our recommendations are relevant for the content of the page, and that our method based on the EQGraph outperforms an alternative baseline based on reverse IR.

The rest of this paper is organized as follows. Section 2 introduces the basic concepts for the EQGraph, which is then

[1] `http://giraph.apache.org`

presented in Section 3. Section 4 describes how we extract entities from pages and queries, how we expand the initial set of entities extracted from a page, and how we produce the suggestions. In Section 5 we assess the quality of the recommendations produced by our method via a user study. Section 6 reviews related literature and Section 7 concludes.

2. PRELIMINARIES

In the following, we first introduce the basic elements of our model, namely, entities, queries, and the Query-Flow graph. Then we describe the EQGraph in the next section.

Entities. We aim at suggesting relevant and interesting queries to a user browsing a web page. To do so, we need a way to effectively represent the *aboutness* of a page, i.e., the set of topics that the page is related to.

The standard approach to this problem requires to perform two tasks: automatic keyword extraction and word sense disambiguation. The former task identifies the sentences in the input text that capture the relevant concepts discussed in the page. The latter task aims at finding the correct interpretation of any such sentence, thus linking it to the actual concept it represents.

Following the state of the art [14, 17, 24, 25, 26], we perform the sense disambiguation task by linking phrases extracted from web pages to Wikipedia articles. We dub *entity* any concept (person, place, event, etc.) that is defined and described in a Wikipedia page. This choice implicitly limits the set of concepts that we can represent in a web page to the ones that have a Wikipedia page describing them.

We believe that Wikipedia provides a fairly comprehensive list of entities of interest. It is nowadays the largest and most visited knowledge repository on the Web, with millions of pages available in many different languages.

Query log. A query log stores information about the *search actions* of the users of a search engine. The basic record stored in a query log $\mathcal{L}$ is a tuple $\langle q_i, u_i, t_i, V_i, C_i \rangle$ where: q_i is the submitted query, u_i is an anonymized identifier for the user who submitted the query, t_i is a timestamp, V_i is the set of documents returned as results to the query, and C_i is the set of documents clicked by the user. In this paper we ignore information from the results of queries (V_i and C_i).

A *user query session*, or *session*, is defined as a sequence of queries of one particular user issued within a given threshold from each other (typically 30 minutes [5]). A *chain* or *logical session*, or *mission*, is a sequence of queries in the same session that are topically coherent.

Query-Flow graph. The Query-Flow graph [5] is a graph representation of a query log aimed at capturing interesting knowledge about the latent querying behavior of the users, with special focus on the sequentiality of similar queries. The nodes of the Query-Flow graph are all the queries in the query log $\mathcal{L}$, and an arc between two queries indicates that the two queries are likely to be part of the same search mission, in the order given by the direction of the arc.

3. THE EQ GRAPH MODEL

We now present our novel graph model, which we call EQGraph. The EQGraph extends the Query-Flow graph [5] with nodes representing Wikipedia entities, and with arcs between entities, as well as from entities to queries.

Bonchi et al. [8] have recently introduced the *Term-Query graph*, which is a Query-Flow graph augmented with term nodes, and with arcs connecting any term to the queries that contain it. The EQGraph is based on a similar principle, but we use Wikipedia entities rather than terms in virtue of their deeper semantics. Entities capture the concepts of interest for people surfing the Web with less ambiguity. They provide a better representation of the content of a page and a more solid base for our query suggestion task.

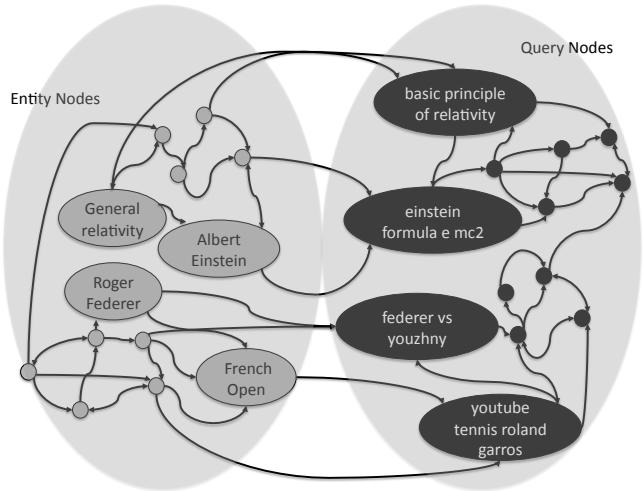


Figure 2: Depiction of the Entity-Query Graph.

More formally, the EQGraph is a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ whose nodes and arcs are defined as follows. Let $\mathcal{Q}$ denote all the queries appearing in a query log $\mathcal{L}$, and $\mathcal{E}$ all the entities extracted from the text of such queries. The set of nodes $\mathcal{N}$ contains a node for every entity $e \in \mathcal{E}$, and for every query $q \in \mathcal{Q}$. Specifically, we denote by $\mathcal{N}_\mathcal{E}$ the set of entity nodes, and by $\mathcal{N}_\mathcal{Q}$ the set of query nodes. Thus $\mathcal{N} = \mathcal{N}_\mathcal{E} \cup \mathcal{N}_\mathcal{Q}$. The set of arcs $\mathcal{A}$ is given by the union of three different sets of arcs: $\mathcal{A}_\mathcal{E}$, $\mathcal{A}_\mathcal{Q}$, and $\mathcal{A}_{\mathcal{E}\mathcal{Q}}$.

Query-to-query transitions. Arcs in $\mathcal{A}_\mathcal{Q}$ connect only query nodes, and they have exactly the same semantic as in the original Query-Flow graph model. An arc $q_i \to q_j \in \mathcal{A}_\mathcal{Q}$, with $q_i, q_j \in \mathcal{N}_\mathcal{Q}$, indicates that the two queries q_i and q_j are likely to be part of the same search mission.

Arc $q_i \to q_j$ is assigned a weight $w_\mathcal{Q}(q_i \to q_j)$ representing the *chaining probability* that a user issues query q_j consecutively after q_i in a same search mission. The chaining probability of any transition $q_i \to q_j$ is computed by extracting temporal, syntactic and session features that aggregate information from all the sessions where q_i and q_j appear consecutively, and in the given order. We discard arcs whose chaining probability is lower than a minimum threshold, because they represent noisy reformulations that are not likely to be made within a given search mission.

The subgraph $(\mathcal{N}_\mathcal{Q}, \mathcal{A}_\mathcal{Q})$ is actually a Query-Flow graph. We follow the settings of the original paper [5] to generate the set of query-to-query arcs $\mathcal{A}_\mathcal{Q}$.

Entity-to-query transitions. To extract entities from queries we use the methodology described in the next section. We denote by $\mathcal{X}_\mathcal{E}(q)$ the set of entities extracted from a query q. We build the set of arcs $\mathcal{A}_{\mathcal{E}\mathcal{Q}}$ by connecting each entity to all the queries that contain it:

$$\mathcal{A}_{\mathcal{E}\mathcal{Q}} = \{e \to q, \text{ s.t. } e \in \mathcal{N}_\mathcal{E}, q \in \mathcal{N}_\mathcal{Q}, e \in \mathcal{X}_\mathcal{E}(q)\}.$$

Any arc $e \rightarrow q \in \mathcal{A}_{\mathcal{EQ}}$ connecting an entity e to a query q is given a weight proportional to the relative frequency of the query q in the log $\mathcal{L}$, with respect to all the queries that contain e. Formally,

$$w_{\mathcal{EQ}}(e \rightarrow q) = \frac{f(q)}{\sum_{q_i | e \in \mathcal{X}_{\mathcal{E}}(q_i)} f(q_i)},$$

where $f(q)$ indicates the frequency of query q in the log $\mathcal{L}$.

Entity-to-entity transitions. Arcs in $\mathcal{A}_{\mathcal{E}}$ connect only entity nodes, and their semantic is intended to be similar to that of arcs in $\mathcal{A}_{\mathcal{Q}}$.

The set of entity nodes $\mathcal{N}_{\mathcal{E}}$ includes all the entities that are contained in the queries in log $\mathcal{L}$. Thus, such entities represent individual concepts of interest, for which people have attempted to gather relevant information by submitting queries to the search engine. In general, an entity can be contained in many different queries, while most of the queries contain just one entity or no entity at all (due to the fact that average query length is very short).

We draw an arc between two entities $e_i \rightarrow e_j \in \mathcal{A}_{\mathcal{E}}$ to indicate that there is high probability that a user who issued one or more queries related to the concept expressed by the first entity e_i, subsequently performs one or more searches related to the concept represented by the second entity e_j.

We build the set $\mathcal{A}_{\mathcal{E}}$ of entity-to-entity arcs starting from the set $\mathcal{A}_{\mathcal{Q}}$ of query-to-query arcs and from the set $\mathcal{A}_{\mathcal{EQ}}$ of the arcs that connect entities to queries.

Given a single transition $q_i \rightarrow q_j \in \mathcal{A}_{\mathcal{Q}}$, let us consider the set of entities $\mathcal{X}_{\mathcal{E}}(q_i)$ extracted from the source query q_i, and the set of entities $\mathcal{X}_{\mathcal{E}}(q_j)$ extracted from the target query q_j. We draw an arc from each entity extracted from the source query, to every entity extracted from the target query. In other words, we derive from $q_i \rightarrow q_j$ a set of transitions $T_{q_i \rightarrow q_j} = \{e_u \rightarrow e_v | e_u \in \mathcal{X}_{\mathcal{E}}(q_i), e_v \in \mathcal{X}_{\mathcal{E}}(q_j)\}$.

Now let $w_{\mathcal{Q}}(q_i \rightarrow q_j)$ be the weight (chaining probability) assigned to arc $q_i \rightarrow q_j$ in our model. With the aim of avoiding to excessively boost the weights assigned to arcs involving popular entities, which may be contained in many queries, we uniformly divide the probability $w_{\mathcal{Q}}(q_i \rightarrow q_j)$ among the $n \cdot m$ entity-entity transitions derived from $q_i \rightarrow q_j$: thus every transition $e_u \rightarrow e_v \in T_{q_i \rightarrow q_j}$ is assigned a probability equal to $p_{q_i \rightarrow q_j}(e_u \rightarrow e_v) = w_{\mathcal{Q}}(q_i \rightarrow q_j)/(n \cdot m)$.

Next, we observe that a given entity-to-entity transition $e_u \rightarrow e_v$ can be derived from multiple query-to-query transitions (all the ones whose source query contains e_u and whose target query contains e_v). Let us assume that there exist r query-to-query transitions $q_{i_s} \rightarrow q_{i_t}$, $i = 1, \ldots, r$ in $\mathcal{A}_{\mathcal{Q}}$ that originate the entity-to-entity transition $e_u \rightarrow e_v$. Let us denote by $p_{q_{i_s} \rightarrow q_{i_t}}(e_u \rightarrow e_v)$, $i = 1, \ldots, r$, the probabilities assigned to transition $e_u \rightarrow e_v$ from each of the r originating $q_{i_s} \rightarrow q_{i_t}$ query-to-query transitions. We aggregate these probabilities to derive the global weight $w_{\mathcal{E}}(e_u \rightarrow e_v)$ that we assign to arc $e_u \rightarrow e_v$ in the EQGraph:

$$w_E(e_u \rightarrow e_v) = 1 - \prod_{i=1,\ldots,r} (1 - p_{q_{i_s} \rightarrow q_{i_t}}(e_u \rightarrow e_v)).$$

Despite the fact that $\mathcal{A}_{\mathcal{Q}}$, which is constructed by following the Query-Flow graph model, does not contain self loops, our procedure for deriving $\mathcal{A}_{\mathcal{E}}$ transitions from $\mathcal{A}_{\mathcal{Q}}$ transitions might lead to the extraction of entity self loops, for example, in the case where the original query-to-query transition is a *specialization*. A specialization [7] is a query reformulation that a user submits when she realizes that her first query was too general given the information need that she had in mind. Thus she formulates a second, more specific query in the attempt of getting a narrower set of results. Typically, the set of words of the specialized query is a superset of the set of words of the first query, thus the entities extracted from the first query are also very likely to be extracted from the second query.

However, we do not include entity self loops in the EQGraph. The scenario for which we introduce our model is that of suggesting relevant queries for web pages, based on the entities they contain. Further in the paper we will show that our recommendation method exploits the entity-entity transitions to perform a preliminary expansion of the initial set of entities extracted from the content of a page. The goal of this entity expansion is to overcome problems that might be encountered when the seed set of entities is particularly poor. We achieve this by performing Personalized PageRank computations on the subgraph of the EQGraph induced by the entity nodes, i.e., the subgraph that contains all the entity nodes, and all the arcs between them. In the computation of Personalized PageRank, self loops could have the effect of boosting the rank of nodes that might not be very close to the preference vector [38].

4. RECOMMENDATION METHOD

We want to suggest to a given user who is browsing a web page p a ranked list Q of k queries such that $q \in Q$ is related to p and *non obvious*.

In this paper we focus on the creation of such a list. We also give a brief overview of the technical aspects of serving these queries, though it is not the main focus of this work.

We decompose the task of finding a set Q of queries that are relevant for a page p into the following subtasks:

1. Extracting a seed set of entities $\mathcal{X}_{\mathcal{E}}(p)$ from p;
2. Expanding $\mathcal{X}_{\mathcal{E}}(p)$ to a larger set of entities $\mathcal{Z}_{\mathcal{E}}(p)$;
3. Obtain queries Q for the expanded set of entities.

Extracting entities from text. To extract entities from a text, which in our framework can be either a web page or a query occurring in a search-engine query log, we first parse the text to identify *surface forms*[2] that are candidate mentions of Wikipedia entities. We add entity candidates to each recognized phrase by retrieving the candidates from an offline Wikipedia database.

To resolve each surface form to the correct Wikipedia entity we apply the machine-learning approach proposed by Zhou et al. [39]. This approach employs a resolution model based on a rich set of both context-sensitive and context-independent features, derived from Wikipedia and various other data sources including web behavioral data.

We then use Paranjpe's *Aboutness* ranking model [26] to rank the obtained Wikipedia entities according to their relevance for the text. This model exploits structural and visual properties of web documents, as well as user feedback derived from search-engine click logs. Paranjpe has shown that his approach, even when trained mainly on head web pages, generalizes and performs well on all kinds of documents, including tail pages.

[2]A surface form is any mention of an entity in the text, e.g., *tomatoes* is a surface form of the entity *Tomato*.

Expanding a list of seed entities. Let p denote a web page for which we want to recommend queries. We perform a Personalized PageRank computation on the graph $\mathcal{G}_\mathcal{E} = (\mathcal{N}_\mathcal{E}, \mathcal{A}_\mathcal{E})$, i.e., the subgraph of the EQGraph that contains all the entities, and all the arcs between them. We use a uniform distribution on the seed set $\mathcal{X}_\mathcal{E}(p)$ as preference vector. We build an expanded set of entities denoted $\mathcal{Z}_\mathcal{E}(p)$ by taking the top entities with the highest scores in the resulting distribution.

Bonchi et al. [8] use a centerpiece subgraph computation on a Term-Query graph rather than Personalized PageRank to recommend queries related to an input query. In the centerpiece computation, the final distribution is given by the Hadamard product of the single seed-node distributions, rather than by their sum as in Personalized PageRank. The choice of [8] is motivated by the fact that they use a term-centric approach. In order to discover queries that are actually relevant for an input query, they have to identify queries that are related to *most* of the terms in the starting query, rather than just to a few of them. As the number of terms in a query is typically very small, this condition is needed to filter out marginally related queries.

In our scenario the centerpiece approach is not a good choice for two reasons. First, the number of seed-entity distributions that we have to aggregate is much larger. In our experiments, we consider pages that contain at least 5 entities, and we extract a maximum of 100 entities per page. With so many starting points, only extremely connected entities would have a non-zero product. Second, entities have a stronger semantic value compared to terms. Terms are more ambiguous and benefit from the disambiguation power of the centerpiece computation. On the contrary, an entity only needs to be relevant enough for some of the seed entities to be considered relevant for a page, and thus to be included in the expanded set. These two facts motivate our choice of Personalized PageRank.

Obtaining queries for entities. Given the expanded set of entities $\mathcal{Z}_\mathcal{E}(p)$ obtained in the previous step, we now want to create a list Q of k related queries to recommend to the user. Our method is similar in spirit to the one considered for the entity expansion step. We still perform a Personalized PageRank computation, but this time on the full EQGraph, by using a uniform distribution over the entities in $\mathcal{Z}_\mathcal{E}(p)$ as preference vector. We return the k query nodes $\{q_1, q_2, \ldots, q_k\}$ that achieve the highest scores in the resulting distribution.

As an alternative method, we implement a *Reverse IR* approach that, given a page p, suggests those queries that have the page in the top positions of their search-engine result list. We represent both queries and pages by means of TF/IDF vectors. Given a page p, we compute the cosine similarity between its TF/IDF vector and the vector of any query stored in the query log. We then return the top k queries that achieve the highest similarity.

5. EXPERIMENTS

5.1 System description

We begin the section with a description of how we envision a system that employs our technique. Our objective is to provide online query recommendations while the user is browsing the Web.

The system consists of two parts: back-end and front-end. The back-end is dedicated to the offline computation of the distributions on the EQGraph. The front-end sends requests to the back-end and shows suggested queries to the user. It can be implemented by a browser plugin or a toolbar add-on.

Workflow. The workflow of our system is as follows. First, the front-end extracts the textual content of the current page and cleans it by using a service such as boilerpipe[3] or by using ad-hoc rules for specific domains. The front-end then sends a request with the cleaned content to the back-end.

The back-end receives the text and extracts relevant entities from it. These entities constitute the seed set for the query suggestion. The back-end then uses this set to query the precomputed index of distributions over the EQGraph, then sums the distributions to obtain the top-k queries to suggest. This operation is very similar to answering a query and can be implemented efficiently on the same infrastructure that a web search engine runs on.

Indexing. The EQGraph has 738 627 entity nodes and 121×10^6 query nodes, so the full distribution would have $738\,627 \times 121 \times 10^6 \approx 89 \times 10^{12}$ entries. Even by compressing the lists the space required to store the full distribution is prohibitive. Thus, following Bonchi et al. [8], we propose to use approximate distributions to make our solution viable. Approximation comes in two ways: pruning and bucketing.

Pruning. By storing only the top-p probabilities for each entity, the space requirements are reduced from $|\mathcal{N}_\mathcal{E}| \times |\mathcal{N}_\mathcal{Q}|$ to $|\mathcal{N}_\mathcal{E}| \times p$. The space saving depends on the condition $p \ll |\mathcal{N}_\mathcal{Q}|$. The price to pay is introducing some error in the query suggested. However, this error is less critical than the one introduced by Bonchi et al. [8] as we use Personalized PageRank rather then centerpiece computation. In Personalized PageRank the final distribution is the sum of the single entity-specific distributions, rather than their Hadamard product. Thus even if one query is missing from the distribution of one of the entities in the seed, it can still be suggested if it is relevant enough to another entity in such set.

Bucketing. Probabilities can be bucketed together by using buckets with exponentially increasing size, as done by Bonchi et al. [8]. We can then save a single probability for the whole bucket, thus greatly reducing the space occupancy. In our case, error propagation is additive rather than multiplicative, so we have a better approximation. See Bonchi et al. [8] for further details. Entity ids in the same bucket can be easily compressed by using gap coding and Elias gamma coding or similar techniques.

5.2 Building the EQGraph

We build a large EQGraph instance from a recent sample of a Yahoo! (US) query log. We anonymize our dataset by discarding all the queries that contain personally identifiable information, and by aggregating information from all the available user query sessions, without retaining any user identifiers. The sample we consider is very large: it contains 24.3×10^9 queries (1.9×10^9 distinct), and 3.3×10^9 transitions between queries.

Normalization. We preprocess the query log to *clean* queries before presenting them to the users. We employ the following query normalization scheme. Given a query q occurring in the log, we remove stop words and non-alphanumeric characters. We then apply Porter's stemming

[3] `http://boilerpipe-web.appspot.com`

Table 1: The EQ Graph: Basic stats

$\|\mathcal{N}\|$	$\|\mathcal{N}_\mathcal{E}\|$	$\|\mathcal{N}_\mathcal{Q}\|$	$\|\mathcal{A}\|$	$\|\mathcal{A}_\mathcal{E}\|$	$\|\mathcal{A}_{\mathcal{E}\mathcal{Q}}\|$	$\|\mathcal{A}_\mathcal{Q}\|$
122 421 398	738 627	121 682 771	202 469 003	17 103 835	40 948 844	144 416 324

Table 2: The EQ Graph: Average degree

$\overline{d_\mathcal{N}}$	$\overline{d_{\mathcal{N}_\mathcal{E}}}$	$\overline{d_{\mathcal{N}_\mathcal{Q}}}$	$\overline{in_\mathcal{N}}$	$\overline{in_{\mathcal{N}_\mathcal{E}}}$	$\overline{in_{\mathcal{N}_\mathcal{Q}}}$	$\overline{out_\mathcal{N}}$	$\overline{out_{\mathcal{N}_\mathcal{E}}}$	$\overline{out_{\mathcal{N}_\mathcal{Q}}}$
3.31	101.75	2.71	1.65	23.15	1.52	1.65	78.59	1.19

algorithm [27] and we sort the stemmed terms lexicographically for each query. As a result we obtain a *normalized* query form $\hat{q}$. This normalization scheme maps multiple queries onto the same normalized form $\hat{q}$ and defines an equivalence class for queries. We thus collapse all the queries that are in the same equivalence class onto the same query. We choose the most frequent query inside each class as the *representative* of the class. Finally, we replace every query in the original log with the representative for its class.

Query graph. We then build $\mathcal{N}_\mathcal{Q}$ and $\mathcal{A}_\mathcal{Q}$ by applying the standard Query-Flow graph methodology [5]. Following the original paper, we filter the log by retaining only the queries that have at least 5 occurrences in the log, and the query-to-query transitions that appear at least 2 times.

Entity graph. From the query log we extract the set of entity nodes $\mathcal{N}_\mathcal{E}$ by following the approach described in Section 4. We then proceed to create $\mathcal{A}_{\mathcal{E}\mathcal{Q}}$ and $\mathcal{A}_\mathcal{E}$. $\mathcal{A}_{\mathcal{E}\mathcal{Q}}$ is obtained simply by connecting every entity in $\mathcal{N}_\mathcal{E}$ to all the queries in $\mathcal{N}_\mathcal{Q}$ that contain it. The set of entity-to-entity transitions $\mathcal{A}_\mathcal{E}$ is derived from $\mathcal{A}_\mathcal{Q}$, using the approach described in Section 3. We discard the top 100 entities with highest in-degree in the graph induced by the entity nodes $\mathcal{G}_\mathcal{E} = (\mathcal{N}_\mathcal{E}, \mathcal{A}_\mathcal{E})$. These entities correspond to popular sites such as Yahoo!, Google, Facebook, Twitter, Youtube, etc. They are mostly derived from navigational queries, and they are not interesting for the purposes of our work.

Altogether we obtain an EQGraph that consists of 122.4×10^6 nodes and 202.4×10^6 arcs . Tables 1, 2, and 3 provide basic statistics about the graph. Observe that the subgraph of the entity nodes is much denser than the subgraph of the query nodes. Figure 3 shows the frequency distribution of queries in the query log, which follows a typical power law.

5.3 Personalized PageRank in Giraph

Given a weighted directed graph, Personalized PageRank is a measure of proximity or relevance of a node to a set of query nodes. To compute this measure on the EQGraph we implement Personalized PageRank on Giraph. Giraph is an open-source large-scale graph-processing framework based on Hadoop. It implements the Bulk Synchronous Parallel model of computation on graphs. Its APIs are inspired by the Pregel system [20].

Our implementation takes as input a directed graph, a restart probability α and a preference vector defined on the nodes of the graph, and generates the stationary distribution of a random walker which restarts from a node in the graph according to the preference vector with probability α.

We use the power iteration method to compute the final

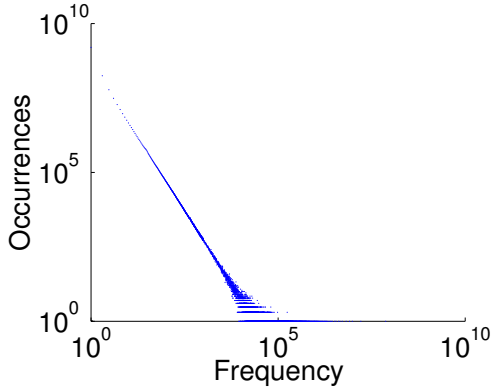


Figure 3: Frequency of queries in the query log

distribution. As standard practice, we set the restart probability $\alpha = 0.15$ and the number of iterations to 30.

In order to reduce the memory requirements of handling the EQGraph, we develop a compact, array-based node representation which allows fast iteration over the arcs but makes random access very expensive. This choice makes sense in the case of PageRank because at each step we need to iterate over all the arcs of each node.

Our implementation of Personalized PageRank can easily scale to very large graphs with billions of nodes and arcs. The code is open source, available online[4], and will be integrated in a future release of Giraph.

5.4 Testing

Our testset is composed of 150 pages drawn randomly from Wikipedia, Yahoo! News, and Yahoo! Finance.

We filter out pages for which we can detect less than 5 entities. We extract the relevant textual content from each page by using a combination of domain specific ad-hoc rules and automatic tools for detecting templates and stripping HTML. Finally, the textual content of each page is fed to our entity detection module to extract entities. These entities constitute the seed set for each page in our experiments.

We have 5 142 unique entities in our testset. As described before, our system would need to perform 5 142 runs of Personalized PageRank, for the entity expansion set. However, for our offline experiments, we leverage the property that PageRank is additive. By setting the preference vector to a uniform distribution over the (expanded) seed set, we need

[4] `https://issues.apache.org/jira/browse/GIRAPH-191`

Table 3: The EQ Graph: Maximum degree

max $d_{\mathcal{N}}$	max $d_{\mathcal{N}_\mathcal{E}}$	max $d_{\mathcal{N}_\mathcal{Q}}$	max $in_{\mathcal{N}}$	max $in_{\mathcal{N}_\mathcal{E}}$	max $in_{\mathcal{N}_\mathcal{Q}}$	max $out_{\mathcal{N}}$	max $out_{\mathcal{N}_\mathcal{E}}$	max $out_{\mathcal{N}_\mathcal{Q}}$
724 634	186 162	724 634	402 121	8 604	402 121	322 513	186 162	322 513

to run 150 Personalized PageRank computations per step, one for each page.

The first round of Personalized PageRank computations produces the expanded seed set $\mathcal{Z}_\mathcal{E}(p)$. We fix $|\mathcal{Z}_\mathcal{E}(p)| = 50$ and for each page p we only keep the top entities in the distribution. This pruning would not be necessary given our offline testing methodology as there is no additional overhead in using all of the entities. However, in a more realistic setting, we would like to reduce the number of Personalized PageRank computations that we need to perform and store. Furthermore, pruning lowly related entities helps to remove noise. Note that the entity expansion step adds entities to the seed set only if the seed set size is less than $|\mathcal{Z}_\mathcal{E}(p)|$.

The second round of Personalized PageRank computations produces the query recommendations. We return the top 5 query suggestions for each page. For all the pages in our testset, we generate recommendations with our EQGraph algorithm, and with the alternative Reverse IR method introduced in section 4. We obtain 1 392 distinct suggestions.

5.5 Evaluation

We ask 10 assessors[5] to evaluate our query recommendations. Each record given to an assessor consists of a (URL, query) pair. The query links to a search-engine result page. The assessor is asked to browse the starting page in order to understand its topic, then to click on the query and read the snippets of the result page to understand the query. Finally, we ask the assessor to rate the relevance of the suggestion on a 3-point scale: "Related and interesting", "Related but obvious", "Unrelated". The following guidelines are provided:

- *Related and interesting*: the query is relevant to the content of the page and expresses a question that cannot be fully answered by reading the web page. The page and the query are related, and you believe that reading the web page would raise your interest in the topics covered by the query.

- *Related but obvious*: the query is equal to the title of the page or the page is top result in the search result page of the query. Page and query are related, but you believe that reading the page would not raise your interest in the query, e.g. the query does not cover any additional aspect with respect to the page content.

- *Unrelated*: the query and the topics of page are clearly unrelated. You believe it is not possible that browsing that page may raise your interest in the query.

The assessors are also given the option of withholding judgment by selecting the "Undecided" option. This choice is designed for difficult cases or when the page is unreadable (e.g. because the page was not available anymore).

Results. Table 4 summarizes the results of our user study. We report aggregated values for the full testset as well as results broken down by domain. We separate the Wikipedia domain from Yahoo! News and Yahoo! Finance because of their intrinsic differences.

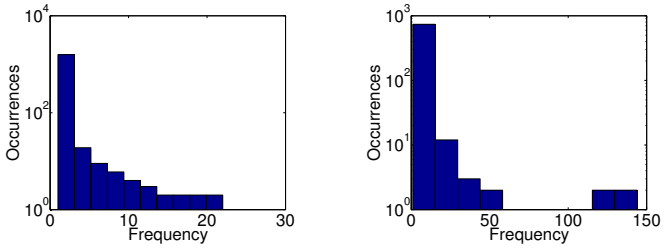
Wikipedia pages are an ideal case for our method: they focus on a single topic but provide many links to continue the exploration, they are very descriptive and easy to understand. They also have a natural affinity with our entity extraction technique that uses Wikipedia pages as entities.

Pages from Yahoo! News and Yahoo! Finance represent a bigger challenge: understanding them is more complex, they present acronyms and jargon that is difficult to decipher, they are also more compact and less descriptive. From technical standpoint, they are less structured and thus harder to clean from HTML tags and templates, so more noise is left in the textual representation of the pages.

EQGraph outperforms Reverse IR in all cases for the "Related and interesting" class. The fraction of interesting suggestions is almost twice as much as the one provided by the baseline for Wikipedia pages (62.7% vs. 33%), and still relevantly higher for the full dataset (58% vs. 36.1%).

As expected, the Reverse IR baseline suggests a large number of obvious queries. This confirms our hypothesis that the problem we are tackling is different from a reverse web search problem. Moreover, we prove that our method addresses exactly the right problem: the number of obvious queries that we suggest to the user is always below 5%.

The price to pay for this *serendipity* is a larger fraction of unrelated queries. Indeed, sometimes the link between a page and a query suggested by the EQGraph is far-fetched and non-obvious. However, this behavior is desired, and in line with research on recommender systems that argues that recommendation is not a prediction task and that accuracy is not the most important metric [21].



(a) EQGraph (b) Reverse IR

Figure 4: Frequency of suggested queries

Diversity. We analyze the results of the query suggestions in terms of diversity. Our methodology does not include any explicit result diversification step. However, by leveraging the EQGraph we implicitly take into account the different topics in each page. Figure 4 shows the histogram of the query frequencies for the results provided by our method and the baseline. The results provided by Reverse IR contain many highly-repeated queries, up to 50 times and beyond (on the x axis). There are also a few popular queries with frequency higher than 100. On the contrary, the results obtained via the EQGraph are more diverse and spread out.

[5]The assessors did not include the authors of this paper.

Table 4: Results obtained with EQGraph, and with Reverse IR baseline

Testset	Label	EQGraph	Reverse IR
Wikipedia pages	Related and interesting	62.7%	33%
	Related but obvious	3.3%	41.5%
	Unrelated	34%	25.5%
Yahoo! News + Yahoo! Finance	Related and interesting	52%	40%
	Related but obvious	2.3%	34.3%
	Unrelated	45.7%	25.7%
Full testset	Related and interesting	58%	36.1%
	Related but obvious	2.9%	38.4%
	Unrelated	39.1%	25.5%

The highest frequency for a single query in this set is 22, while most of the queries are singleton (note the logarithmic scale on the y axis).

We further explore the diversity of the result sets via lexical analysis. Given the result set returned for a page by a given method, we compute the Levenshtein distance between all of the result pairs in the set. We define the lexical diversity $\mathcal{D}$ of a result set for a page as the sum of all the $\frac{n\times(n-1)}{2}$ distances:

$$\mathcal{D}(p) = \sum_{i,j} \text{Lev}(q_i, q_j) \quad \text{for } i,j \in [1,k] \wedge i < j$$

where q_i, q_j are the queries in the result set of page p and k is the cardinality of the set.

We compute the lexical diversity for each page in our testset and for each method. Then we do a statistical analysis of the distribution of the lexical diversity. We apply a dependent t-test for paired samples to pairs of lexical diversity coming from different methods applied to the same page.

The test shows that the lexical diversity of the queries obtained via the EQGraph is higher than the baseline, and is statistically significant with a confidence level of 0.1%.

Anecdotal evidence. We present example suggestions created by both methods in Table 5. We select examples from Wikipedia as they are easier to understand and exemplify at best the kind of interesting queries we want to suggest.

The queries suggested by Reverse IR almost invariably contain the title of the page in their formulation. For the few queries that are in the form of a question, the page itself contains the answer.

On the contrary, the queries suggested by the EQGraph are mostly about specific aspects related to the main topic of the page. For example the role of osteoblasts and osteoclasts (cells that create and destroy bone tissue) in the process of osteoporosis, a bone disease. Or in the case of Azotemia (a condition of high level of nitrogen compounds like urea and creatinine in the blood), about its possible causes and symptoms such as kidney stones and osmotic diuresis.

6. RELATED WORK

Implicit and Context-Aware Search Predictions. Cheng *et al.* [10] actively predict search intents from user browsing behavior data. The basic idea is that if, after visiting a particular page, many users often continue with the same query then the original page is likely to "trigger" the query. To predict such queries, they learn a model to effectively rank such follow-up queries according to their likelihood of being triggered by the page. They also propose an approximation algorithm to diversify the ranked lists of queries obtained. They test their approach on large-scale user browsing behavior data obtained from a commercial search engine. Though their basic problem, identifying queries likely to be "triggered" by a visited web page, is the same as ours, our work differs significantly. Their approach requires historic browsing information and is not appropriate for previously unseen pages. They focus on identifying relevant (page, query) transitions from an existing candidate set whereas we, with tail pages in mind, focus on the creation of such sets in the first place.

The authors in [19] predict the next query in a session, using both search and click information. Though many of their features used are not applicable for previously unseen URLs, some, such as the DMOZ category of pages visited, are. In their experiments the best features are the previous query and the clicked URL. Their approach is largely orthogonal to ours. We focus on content-based techniques, not on log-based ones. Also, the problem they study has a crucial difference: for predicting the next query, "obvious" queries – often corresponding to repeat queries [31, 33] – are actually good, whereas they are bad for our purposes.

Cao *et al.* [9] propose a hidden markov model for predicting user behavior in a search session. Given a user's session so far, the model can predict the next query, and the urls the user is likely to click. The hidden states in the model describe the user's intent, while observed variables are the queries, and clicks on urls. The probability distributions over queries and urls depend only on the current state, but the state transitions depend on all previous states of the session. Similar to the work in [10] this approach could work in our setting *given enough observed page-to-query transitions*. However, the method fails to suggest queries while browsing *tail* pages, while our approach is not affected by this.

Finally, the problem of query prediction has been studied in the context of *caching* [18]. In caching, the focus is less on the user and more on the system. E.g., the system might benefit from knowing which queries are currently trending to cache their results, but such queries are unlikely to be of use for the problem studied by us.

Query suggestions. There is a large body of work on the problem of query suggestions in the context of web search [22, 5, 15]. Though such work is clearly related, note that our problem is not one of helping users to formulate their *existing* information need. Rather we are interested in possibilities to *create* such needs.

Work on query suggestions that is more closely related to our paper includes [23]. There the authors employ "semantic" information from dbpedia for the suggestions, which at a high level is related to our use of Wikipedia entities. Similarly, the work in [11] is related because the authors also

Table 5: Query suggestions for some Wikipedia pages

Page	EQGraph	Reverse IR
Medulla Oblongata	the pyramidal decussation occurs in the	difference between medulla and medulla oblongata
	location of middle cerebellar peduncles	what does the medulla do
	somatic motor nuclei	medulla
	bilateral middle cerebellar peduncle lesion	medulla oblongata
	cortical tissue running between the medullary pyramids	parts of medulla oblongata
Osteoporosis	function of osteoblasts	bone osteoporosis
	osteoclast and osteoblast	osteoporosis
	vasovagal syncope	medicinenet.com osteoporosis
	symptoms of pulmonary embolism	mayoclinic.com osteoporosis
	partial cauda equina syndrome	osteoporosis fractures
Azotemia	symptoms of kidney stones	creatinine
	coffee and diabetes	why could a creatinine would only go up
	fluid volume deficit related to osmotic diuresis	what is creatinine for
	signs and symptoms of osmotic diuresis	what does creatinine do
	blood urea nitrogen	medicinenet.com creatinine

use *content* of clicked pages in constructing query suggestions, differing from most of the other work in this area. As in our paper, a random walk on a graph model where nodes are queries is applied in [6, 5]. Finally, the work in [30] is related due to its focus on query suggestions for infrequent, long-tail queries. The authors propose the use of *templates* to learn rules such as "[CITY] flight" -> "[CITY] hotel". These rules can then be applied for previously *unseen* queries such as "bertinoro flight -> bertinoro hotel", both as input and output, as long as "bertinoro" is identified as a city. In future work, we will consider a similar approach in our setting: if after reading about a person X the query "X place of birth" is frequently suggested by our system, we could also suggest such a query, even when it is previously unseen for a concrete person Y.

User intent modeling. At a high level, our work is also related to user intent modeling. Note, however, that unlike in search-result ranking our suggestions make an effort to *influence* a user's intent, rather than merely guiding him.

Shen *et al.* [28] focus on the problem of exploiting implicit user feedback to improve the accuracy of a retrieval feedback via context-sensitive language models that combine the current query with the interaction history (previous queries and clicks). Again, our aim is that of *suggesting* the next query, not combining its signal with previous relevance signals.

Agichtein *et al.* [1] show how to incorporate "user behavior" to improve web search. They discuss how to merge two distinct ranked lists, how to re-rank results or how to incorporate the new features in a learning to rank model. Somewhat relevant to our problem is the type of features they use. Apart from traditional CTR information they use information such as "are results below this one ever clicked on the search result page", "is this result typically found via links", or "what's the average dwell time on this web page". In our setting, such features fail for previously unseen pages.

Zhu *et al.* [40] expand the use of browsing information for web search ranking, and other applications. They introduce *ClickRank*, an algorithm to estimate web page importance from browsing information. Finally, [40] discusses novel applications of ClickRank in providing enriched user web search experience.

White *et al.* [36] present a web search interaction feature which, for a given query, provides links to web sites frequently visited by other users with similar information needs. These "popular" destinations complement traditional search results, allowing direct navigation to authoritative resources. Results show that search enhanced-by-destination suggestions outperform other approaches. In a certain sense such "suggestions", rather than search results, are dual to our problem: given a page we want to suggest queries, whereas they suggest pages for a given query.

White and Chandrasekar [35] present a comparative oracle study of techniques to shortcut sub-optimal search trails using labels derived from social bookmarking, anchor text, query logs, and a human-computation game. The authors show that labels can help users reach target pages efficiently, and that shortcuts are most useful when the target is challenging to find. As such shortcuts are derived from *observed* browsing behavior, similar approach would not be applicable for our problem where we are also addressing tail pages.

Xiang *et al.* [37] investigate how to improve web-search results by taking into account "context" information. For example, if a user searches for 'BMW' and then for 'jaguar', it is likely that he is interested in the car rather than the cat. In this work "context" always means the previous query. So a user submits query q_{t-1}, clicks on some results, then submits query q_t. In our setting, one could try modeling the currently browsed page as q_{t-1} and use a similar approach to theirs, but they rely on observing transitions from q_{t-1} to q_t where we want to *suggest* such transitions.

On the general topic of query-intent modeling, [12] and [29] are related as they, respectively, present novel methods for query clustering and query intent classification that leverage user interaction logs.

7. CONCLUSIONS

Suggesting a small and diverse set of search queries that are relevant to the content of the web page currently visited by a user, and that moreover are non-obvious and serendipitous, is an interesting problem which has received (so far) surprisingly little attention. In this paper we tackle such a problem focusing on the content of the page and the entities contained therein. Our proposal is based on a novel graph model which enriches the Query-Flow graph with entity nodes. Our model is content-based and can produce meaningful recommendations even for previously unseen pages.

Our recommendation method is based on a two-step Personalized PageRank computation, whose goal is to expand the set of entities extracted from a page into a richer set of entities, and to associate the entities with relevant query suggestions. We develop an efficient implementation that enables our framework to deal with very large graph instances and suggest queries from a large and diverse pool.

Our experiments show that our recommendations are relevant and serendipitous, outperforming in a user study, a method based on reverse IR. As our method is tailored towards avoiding related-but-obvious recommendation, we sometimes produce some unrelated suggestions. These are easy to identify and can be filtered out by a specialized classifier: we leave this further development for future work, as well as evaluating other expansion strategies and the effect of parameters in the quality of recommendations.

8. ACKNOWLEDGEMENTS

This work was partially supported by the Torres Quevedo Program from the Spanish Ministry of Science and Innovation, co-funded by the European Social Fund.

References

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pp. 19–26, 2006.

[2] P. André, m. schraefel, J. Teevan, and S. T. Dumais. Discovery is never by chance: designing for (un)serendipity. In *C&C*, pp. 305–314, 2009.

[3] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *KDD*, pp. 76–85, 2007.

[4] R. Baeza-Yates, C. Hurtado, and M. Mendoza. *Query Recommendation Using Query Logs in Search Engines*, volume 3268, pp. 588–596. Springer, 2004.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM*, pp. 609–618, 2008.

[6] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *WSCD*, pp. 56–63, 2009.

[7] 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 Press, 2009.

[8] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. Efficient query recommendations in the long tail via center-piece subgraphs. In *SIGIR*, p. to appear, 2012.

[9] H. Cao, D. Jiang, J. Pei, E. Chen, and H. Li. Towards context-aware search by learning a very large variable length hidden markov model from search logs. In *WWW*, pp. 191–200, 2009.

[10] Z. Cheng, B. Gao, and T.-Y. Liu. Actively predicting diverse search intent from user browsing behaviors. In *WWW*, pp. 221–230, 2010.

[11] S. Cucerzan and R. W. White. Query suggestion based on user landing pages. In *SIGIR*, pp. 875–876, 2007.

[12] J. Cui, H. Liu, J. Yan, L. Ji, R. Jin, J. He, Y. Gu, Z. Chen, and X. Du. Multi-view random walk framework for search task discovery from click-through log. In *CIKM*, pp. 135–140, 2011.

[13] B. M. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani. Concept-based interactive query expansion. In *CIKM*, pp. 696–703, 2005.

[14] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *EMNLP*, pp. 782–792, 2011.

[15] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, pp. 387–396, 2006.

[16] R. Kop. The Unexpected Connection: Serendipity and Human Mediation in Networked Learning. *Educational Technology and Society*, pp. 2–11, 2012.

[17] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in web text. In *KDD*, pp. 457–466, 2009.

[18] R. Lempel and S. Moran. Predictive caching and prefetching of query results in search engines. In *WWW*, pp. 19–28, 2003.

[19] K. H.-Y. Lin, C.-J. Wang, and H.-H. Chen. Predicting next search actions with search engine query logs. In *WI*, pp. 227–234, 2011.

[20] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A System for Large-Scale Graph Processing. In *SIGMOD*, pp. 135–145, 2010.

[21] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI*, pp. 1097–1101, 2006.

[22] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, pp. 469–478, 2008.

[23] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. de Rijke. Learning semantic query suggestions. In *ISWC*, pp. 424–440, 2009.

[24] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pp. 233–242, 2007.

[25] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *CIKM*, pp. 509–518, 2008.

[26] D. Paranjpe. Learning document aboutness from implicit user feedback and document structure. In *CIKM*, 2009.

[27] M. F. Porter. *An algorithm for suffix stripping*, pp. 313–316. Morgan Kaufmann Publishers Inc., 1997.

[28] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR*, pp. 43–50, 2005.

[29] Y. Shen, J. Yan, S. Yan, L. Ji, N. Liu, and Z. Chen. Sparse hidden-dynamics conditional random fields for user intent understanding. In *WWW*, pp. 7–16, 2011.

[30] I. Szpektor, A. Gionis, and Y. Maarek. Improving recommendation for long-tail queries via templates. In *WWW*, pp. 47–56, 2011.

[31] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: repeat queries in Yahoo's logs. In *Proc. SIGIR'07*. ACM, 2007.

[32] E. Toms. Information exploration of the third kind: the concept of chance encounters. In *Information Exploration Interfaces @ CHI*, 1998.

[33] S. K. Tyler and J. Teevan. Large scale query log analysis of re-finding. In *WSDM*, pp. 191–200, 2010.

[34] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[35] R. W. White and R. Chandrasekar. Exploring the use of labels to shortcut search trails. In *SIGIR*, pp. 811–812, 2010.

[36] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *SIGIR*, pp. 159–166, 2007.

[37] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *SIGIR*, pp. 451–458, 2010.

[38] M. Yazdani and A. Popescu-Belis. Using a Wikipedia-based Semantic Relatedness Measure for Document Clustering. In *TextGraphs*, pp. 29–36, 2011.

[39] Y. Zhou, L. Nie, O. Rouhani-Kalleh, F. Vasile, and S. Gaffney. Resolving surface forms to Wikipedia topics. In *COLING*, pp. 1335–1343, 2010.

[40] G. Zhu and G. Mishne. Mining rich session context to improve web search. In *KDD*, pp. 1037–1046, 2009.