When Entities Meet Query Recommender Systems: Semantic Search Shortcuts

Diego Ceccarelli[1,2,3] Sergiu Gordea[4] Claudio Lucchese[1]
Franco Maria Nardini[1] Raffale Perego[1]

[1] ISTI-CNR, Pisa, Italy – {firstname.lastname}@isti.cnr.it
[2] IMT Institute for Advanced Studies Lucca, Lucca, Italy
[3] Dipartimento di Informatica, Università di Pisa, Pisa, Italy
[4] AIT GmbH, Wien, Austria – sergiu.gordea@ait.ac.at

ABSTRACT

The Web of Data is growing in popularity and dimension, and entities are gaining importance in many research fields. In this paper, we explore the use of entities that can be extracted from a query log to enhance query recommendation. In particular, we use a large query log recorded by the Europeana portal, a central access point to the descriptions of more than 20 million cultural heritage objects, and we extend a state-of-the-art query recommendation algorithm to take into account the semantic information associated with the submitted queries. Our novel method generates highly related and diversified suggestions. We assess it by means of a new evaluation technique. The manually annotated dataset used for performance comparisons has been made available to the research community to favor the repeatability of the experiments.

Categories and Subject Descriptors

H.3.3 [**Information Process and Retrieval**]: Search Process

Keywords

Semantic Query Suggestion; Search Shortcuts

1. INTRODUCTION

The strong inclination for culture and beauty in Europe created invaluable artifacts starting from antiquity up to nowadays. That cultural strength is recognized by all people in the world and it makes Europe the destination for a half of the international tourists[1].

[1]http://www.unwto.org/facts/eng/pdf/highlights/UNWTO_Highlights10_en_HR.pdf

The Europeana digital library launched its public Web portal[2] in autumn 2008. It provides a central access point to the description of more than 20 million cultural heritage objects, available in a dataset that has been recently made available for reuse[3]. Due to the large amount of information published within the portal, the access to the description of a specific masterpiece becomes a challenging task, given the fact that each object is described in one of the 28 languages and dialects by using one of the three major groups of European typefaces (i.e. Latin, Greek, Russian).

In this paper, we present an approach that aims at supporting users to easier find the information they need, by providing recommendations for good search queries and by reusing the knowledge provided by users when interacting with the search engine.

Search query recommendation techniques [1, 6, 9] are commonly used in web search engines to help users refining their queries. These technologies analyze the user behavior by mining the system logs in order to find the correlation between the user's information need (i.e. what? - visited pages), what the user is searching for (i.e. how? - query terms) and the content and structure of the information pool (i.e. search index).

To enhance the information conveyed by query recommendations and to compensate the weaknesses of plain query suggestions, we present an approach that improves these suggestions by identifying and exploring their semantic information. The contribution of the paper is threefold:

- we propose Semantic Search Shortcuts (S^3): a query recommender system that exploits named entities associated with users' query to improve the underlying recommendation model and provide meaningful suggestions;
- we propose two general evaluation metrics that measure relatedness and diversity of suggestions by using the named entities that could be associated to each query.
- we propose a manually annotated dataset in which we assign named entities (i.e. Wikipedia resources) to a set of search queries and their related suggestions. This dataset has been made available for download to sup-

[2]http://www.europeana.eu/portal
[3]http://www.pro.europeana.eu/web/guest/re-use-data

port further experimentations of the scientific community;

The rest of the paper is organized as follows: Section 2 presents the work related to our approach, while Section 3 describes the novel semantic query recommendation system. In Section 4, we describe the evaluation methodology and we discuss our experimental results. Finally, in Section 5, we draw conclusions and outline some future work.

2. RELATED WORK

Our research addresses two main topics, namely, the recommendation of queries and the identification of named entities.

The linking of culture related texts with entities from the web of linked data is realized by using semantic annotations [15, 23, 25]. The entity detection became an important research topic in this context. Han *et al.* [15] introduce the concept of collective entity linking which exploits the interdependency between entities when annotating webpages. Similar to their approach, we are using the relationships between entities for improving the accuracy of semantic search queries. Mihalcea and Csomai present the WikiFy! system[23] which was build to annotate textual descriptions or regular web pages with the concepts available in Wikipedia.

Meij *et al.* [22] addresses also the suggestion of topics related to cultural heritage domain. The authors analyze the query logs of the Dutch Sound and Vision Institute and use machine learning algorithms for identifying the named entities related to the search queries. Within their approach, the authors are searching the Dutch Wikipedia for identifying entities related to the search queries. Basically, they are using the external resources to generate a context of the search query. In contrast to their approach, we are generating the context basing on the search results of Europeana, which we believe that is better related to the information searched by users.

Query suggestion techniques address specifically the problem of recommending queries to Web search engine users, and propose specific solutions and specific evaluation metrics tailored to the Web search domain. Techniques proposed during last years are very different, yet they have in common the exploitation of usage information recorded in query logs [27]. Many approaches extract the information used from the plain set of queries recorded in the log, although there are several works that take into account the chains of queries that belong to the same search session [26]. In the first category we have techniques that employ clustering algorithms to determine groups of related queries that lead users to similar documents [1, 5, 28]. The most "*representative*" queries in the clusters are then returned as suggestions. Others solutions employ the reformulations of the submitted query issued by previous users [17], or propose as suggestions the frequent queries that lead in the past users to retrieve similar results [3].

Among others, Baeza-Yates *et al.* exploit click-through data as a way to provide recommendations [2]. The method is based on the concept of *Cover Graph*. A Cover Graph is a bipartite graph of queries and URLs, where a query q and an URL u are connected if a user issued q and clicked on u that was an answer for the query. Suggestions for a query q are thus obtained by accessing the corresponding node in the Cover Graph and by extracting the related queries sharing more URLs. The sharing of clicked URLs results to be very effective for devising related queries, and the Cover Graph solution has been chosen as one of the two query suggestion algorithms considered in this paper for experimental performance comparison.

Boldi *et al.* introduce the concept of *Query Flow Graph*, an aggregated representation of the information contained in a query log [6]. A Query Flow Graph is a directed graph in which nodes are queries, and the edge connecting node q_1 to q_2 is weighted by the probability that users issue query q_2 after issuing q_1. Authors highlight the utility of the model in two concrete applications, namely, *devising logical sessions* and *generating query recommendation*. The authors refine the previous studies in [7] and [8] where a query suggestion scheme based on a random walk with restart model on the Query Flow Graph is proposed.

Downey *et al.* describe search log studies aiming at explaining behaviors associated with rare and common queries [13]. They investigate the search behavior following the input of rare and common queries. Results show that search engines perform less well on rare queries. The authors also study transitions between rare and common queries highlighting the difference between the frequency of queries and their related information needs.

Mei *et al.* propose a novel query suggestion algorithm based on ranking queries with the hitting time on a large scale bipartite graph [21]. The rationale of the method is to capture semantic consistency between the suggested queries and the original query. Empirical results on a query log from a real world search engine show that hitting time is effective to generate semantically consistent query suggestions. The authors show that the proposed method and its variations are able to boost long tail queries, and personalized query suggestion. Kang *et al.* retrieve entities from several sources (query logs, Flickr, Wikipedia) and use a gradient boosting algorithm to obtain a function which predicts the relative relevance between two entities based on their concurrence [18].

Broder *et al.* propose to leverage the results from search engines as an external knowledge base for building the word features for rare queries [11]. The authors train a classifier on a commercial taxonomy consisting of 6,000 nodes for categorization. Results show a significant boost in term of precision with respect to the baseline query expansion methods. Lately, Broder *et al.* propose an efficient and effective approach for matching ads against rare queries [10].

3. SEMANTIC QUERY RECOMMENDATION

The typical interaction of a user with a Web search engine consists in *translating her information need in a textual query made of few terms.* Such queries are usually stored by the search system in its *query log.* Mining query logs to study the users' past interactions is an effective approach improve the efficacy of a search system, and in particular to produce relevant query suggestions. This is based on the assumption that successful search activity by previous users can be of interest for others. We believe that the "*Web of Data*" can be profitably exploited in the query log mining process to alleviate possible vocabulary mismatch problems, and that it can be leveraged to provide more more user-friendly query suggestions.

3.1 The Search Shortcuts Model

We adopt the *Search Shortcuts* (SS) model proposed in [4, 9] and its terminology. Let $\mathcal{Q}$ be the set of queries recorded in a query log by the set of users $\mathcal{U}$. The query log is preprocessed by using some session splitting method (e.g., [16, 20]) in order to extract query *sessions*, i.e., sequences of queries which are related to the same user search task. Formally, we denote by $\mathcal{S}$ the set of all sessions in the query log. A session $\sigma \in S$ is composed by a set of queries $\sigma = \{\sigma_1, \ldots, \sigma_n\}$. A session $\sigma \in S$ is said to be *successful* if and only if the user has clicked one of the links shown in the result page of the final query σ_n, and *unsuccessful* otherwise.

The SS algorithm aims to generate suggestions containing only those queries appearing as final in successful sessions. The goal is to suggest queries that have already been proved to be successful for some users to achieve their own goal. The SS algorithm works by efficiently computing the similarity between currently session of a given user and the historical successful sessions recorded in a query log. Final queries of most similar successful sessions, named **search shortcuts**, are suggested to the user.

The idea described above can be translated into the following process. For each unique *final query* contained in successful sessions we define a *virtual document* $\mathcal{VD}$ identified by its *title* and its *content*. The title identifies uniquely a $\mathcal{VD}$ and it is actually the final query. The content of the virtual document is instead composed of all the terms that appeared in queries of all the successful sessions ending with their final query. At the end of this procedure for each successful final query q, we have a $\mathcal{VD}_q$ containing all the queries appearing in the sessions that end with q.

Example: Consider these two successful sessions: (*dante alighieri* $\rightarrow$ *divina commedia* $\rightarrow$ *paolo e francesca*), and (*divina commedia* $\rightarrow$ *inferno canto V* $\rightarrow$ *paolo e francesca*). We create the virtual document identified by the title *paolo e francesca* and whose content is the text (*dante alighieri divina commedia divina commedia inferno canto V*). The virtual document actually contains also repetitions of the same terms. All virtual documents are indexed with an Information Retrieval system. Generating shortcuts for a given user query q is a matter of processing the query q over the inverted file indexing such virtual documents. The titles of the top-k most similar virtual document retrieved are finally returned as suggestions. We use a TF–IDF weighting to score virtual documents w.r.t. the given query. Processing queries over inverted indexes is very fast and scalable, and these important characteristics are inherited by this query suggestion technique as well.

Another important feature of the SS query recommender system is its robustness with respect to rare and singleton queries. Singleton queries account for almost 50% of the submitted queries, and their presence causes the issue of the sparsity of models [27]. Since the query is matched on the text obtained by concatenating all the queries in each session, the system does not need to find an identical previously submitted query as in the case of other suggestion algorithms. Therefore, it is able to generate suggestions for rare queries, whose terms do have context information in the query log used to build the model.

3.2 Hidden Knowledge in Search Sessions

A search session is an interactive process where users continuously refine their queries in order to better specify their information need. Sometimes, the successful query is not known in advance, but users might adopt concepts and terminologies also on the basis of the results pages visited. The approach described here, exploits successful queries from successful sessions to recommend queries that allowed "*similar*" users, i.e., users which in the past followed a similar search process, to successfully find the information they were looking for, and it is able to catch non trivial semantic relationships among queries.

Even if query logs hide precious knowledge that can be profitably exploited for many applications, they contain a lot of noise. In the Europeana query log, several kinds of noise could be identified: i) multiple representation of the same entity, due to different languages (e.g., en/Divine Comedy vs. it/Divina Comedia), or different way to denote the same entity (e.g., *leonardo*, or *da vinci*, are both referring to the painter *Leonardo Da Vinci*), ii) homonyms (e.g., the term *war* refers to several different historical events), iii) multi-goal search sessions, when users search for multiple loosely connected information pieces in the same search session (e.g. search for *Inferno* and for *Gioconda*). In order to be able to provide good search terms recommendations it is mandatory to remove most of the noise from the recommendation index.

These problems are well known within *annotation systems* literature [14, 15, 19, 24]. An annotation system is a framework that takes a text as input and returns a set of subsequences (called *spots*) within the text, each one referring to an entity. This matching problem is usually solved in two steps: the *spotting*, in which a set of candidate spots is selected, and the *disambiguation*, in which each spot is tagged with the most relevant entity chosen among the associated candidates.

Spotting and disambiguation are performed by exploiting a knowledge base (KB) containing a collection of entities $KB = \{e_1, e_2, \cdots, e_n\}$. Each entity belonging to the knowledge base is identified by a unique id, and has several attributes. For example a set of spots that could be used to refer to the entity, a textual description of the entity, its *type* (e.g., person, museum, book, ...), the set of *semantic relations* with other entities in the KB, etc.

A popular KB used for annotating texts is Wikipedia. In this case, each article (excluding redirect/disambiguation pages and categories) can be considered an entity. Moreover, a large part of the articles contains **infoboxes**, a fixed-format table appearing in the wikipedia page from which attributes can be easily extracted. Finally, spots can be obtained by considering the anchors of all the links pointing to the entity, and semantic relations by hyperlinks.

3.3 Semantic Search Shortcuts

In order to perform the mapping between a query and its associated entities, we preliminary annotate query sessions and virtual documents. Most query annotation approaches consider single queries and try to map them to an entity. If a query is ambiguous, the risk is to always map it to the most popular associated entity. On the other hand, by performing the mapping by using all the queries in a user session, we can exploit other queries as a source of contextual information and improve precision remarkably.

The Collective Entity Linking approach proposed by Han *et al.* [15] was implemented by using a recent dump of English Wikipedia. We direct interested readers to the original paper for a detailed explanation of the annotation method.

The dump contains 4,677,051 entities, denoted by 11,318,080 distinct spots, and interconnected by 104,220,848 semantic relations. It is worth to observe that we used only the English language part of Wikipedia. Even though queries in the query log are written in different languages we decided to use the English version of Wikipedia as it contains a large number of spots in other languages, in form of anchors or redirects (e.g., Italian *Divina Commedia* is redirected to *Divine Comedy*). We leave the investigation regarding the use of different languages as future work.

By means of the Collective Entity Linking annotator we tagged the text of each virtual document $\mathcal{VD}$. More precisely, we keep track of each session belonging to the content of the virtual documents, and we used the annotator to identify mappings between queries and entities. Each identified entity is then added to a new field *entities* of the virtual document. The annotator is a complex machine learning-based tool that also provides a confidence value of every annotation proposed. We store also such confidence value in the index. At the end of the process, our inverted index contains a set of virtual documents, each one consisting of three fields: *title*, *content*, and *entities*. We observe that by following this approach, if the title of a given virtual document is affected by homonymy, the entities field is likely to contain disambiguations representing some possible meanings. We define *Semantic Search Shortcuts* (S^3) the query recommender system exploiting this new added knowledge that allows us to suggest queries by exploiting the semantic relations among the current query and those submitted in the past. Note that, differently from traditional recommenders that generate for each query a flat list of recommendations, S^3 provides a list of related entities. Suggesting entities instead of textual queries can be convenient for a number of reasons. First entities do not suffer from language issues. Moreover, the enriched semantic of S^3 recommendations enables a more user friendly representation within the portal. For example, some kind of suggested entities (e.g., paintings, people) can be more clearly represented by images.

Given an input query q, in order to compute the entities to be suggested we first retrieve the top-k most relevant virtual documents by processing the query over the SS inverted index built as described in Section 3.1. The result set R_q contains the top-k relevant virtual documents along with its associated entities. Given an entity e in the result set, we define:

$$score(e, \mathcal{VD}) = \begin{cases} conf(e) \times score(\mathcal{VD}), & \text{if } e \in \mathcal{VD}.entities \\ 0 & \text{otherwise} \end{cases}$$

where $conf(e)$ is the confidence of the annotator in mapping the entity e in the virtual document $\mathcal{VD}$, while $score(\mathcal{VD})$ represents the similarity score returned by the information retrieval system. We then rank the entities appearing in R_q using their score w.r.t. the query, as below:

$$score(e, q) = \sum_{\mathcal{VD} \in R_q} score(e, \mathcal{VD})$$

4. EXPERIMENTAL EVALUATION

We use a large query log coming from the Europeana portal[4], containing a sample of users' interactions covering two years (from August 27, 2010 to January, 17, 2012). An extensive characterization of an Europeana query log can be found in [12]. We preprocess the entire query log to remove noise (e.g., queries submitted by software robots, mispells, different encodings, etc). We then apply a time-based session splitting technique. We use a threshold of 30 seconds between each consecutive query pair. By doing so, we obtain 139,562 *successful* sessions. Finally, we use these successful session to build the inverted index as described in Section 3.1.

4.1 Relatedness and Diversity

We are interested in evaluating two aspects of the set of suggestions provided. These are our main research questions:

Relatedness : How much information related to the original query a set of suggestions is able to to provide?

Diversity : How many different aspects of the original query a set of suggestions is able to cover?

In order to evaluate these aspects we borrow from the annotators the concept of *semantic relatedness* between two entities proposed by Milne and Witten [24]:

$$rel(e_1, e_2) = 1 - \frac{log(max(|I_L(e_1)|, |I_L(e_2)|)) - log(|I_L(e_1) \cap I_L(e_2)|)}{log(|KB|) - log(min(|I_L(e_1)|, |I_L(e_2)|))}$$

where e_1 and e_2 are the two entities of interest, the function $I_L(e)$ returns the set of all entities that link to the entity e in Wikipedia, and KB is the whole set of entities in the knowledge base. We extend this measure to compute the similarity between two set of entities (the function I_L gets a set of entities and returns all the entities that link *at least* on entity in the given set). At the same time, given two sets of entities E_1, E_2, we define the diversity as $div(E_1, E_2) = 1 - rel(E_1, E_2)$. Given a query q, let E_q be the set of entities that have been manually associated with the query. We define the relatedness and the diversity of a list of suggestions S_q as:

Definition 1. *The average relatedness of a list of suggestions is computed as:*

$$rel(S_q) = \frac{\sum_{s \in S_q} rel(E_s \setminus E_q, E_q)}{|S_q|}$$

where E_s represents the set of entities mapped to a suggestion s (could contain more than one entity in the manual annotated dataset). Please note that we remove the entities of the original query from each set of suggestions as we are not interested in suggesting something that do not add useful content w.r.t. the starting query ($E_s \setminus E_q$).

Definition 2. *The average diversity of a list of suggestions is defined as:*

$$div(S_q) = \frac{\sum_{s \in S_q} div(E_s, E_{S_q \setminus s})}{|S_q|}$$

[4] http://www.europeana.eu/portal/

For each suggestion, we want to evaluate how much information it adds w.r.t. the other suggestions. $E_{S_q \setminus s}$ denotes the union of the entities belonging to all the suggestions except the current suggestion s.

Note that diversity and relatedness are very often two contrasting objectives. In order to maximize relatedness, and algorithm should provide the results being most related with the user query. On the other hand, such results are likely to be very similar to each other. The correct trade-off between relatedness and diversity would be able to provide high quality results covering multiple topics being relevant for the user query.

4.2 Evaluation Methodology

The evaluation of query recommendation techniques is an open issue. In most cases, it is addressed by performing adhoc user studies. User studies are time-consuming activities, very difficult to be repeated for comparative evaluations of different methods. In this work, we intend to overcome the problem by proposing a new way to evaluate query recommendation. In particular, we want to assess how query recommender systems performs in terms of entities suggested thus enabling us to use the metrics (i.e., relatedness and diversity) defined above. We built a dataset consisting in 130 queries split in three disjoint sets. The 50 queries in the first set are short (composed by only one term). The second set contains 50 queries having an average length of 4.2 terms, while the 30 queries in the third set have an average length of 9.93 terms. For each query in the three sets, we compute the top-10 recommendations produced by the SS query recommender system and we map them to entities by using a simple interface providing an user-friendly way to associate entities to queries. Three annotators participated to the manual annotation. They manually annotated queries and their related recommendations with one or more Wikipedia entities. An entity is represented by its *title* and its numerical id, available in a recent English Wikipedia dump. The whole set of queries and recommendations has been randomly divided in three distinct sets. Each of those sets has been annotated by one annotator and cross-checked by the other two.

The rationale of using different sets of queries varying for their length is that we want to assess how our technique performs (in terms of entities suggested) w.r.t. the length of the query. Short queries are, in fact, easier to manage as they usually represent a single entity. Longer queries are more difficult as they may be associated with more than one entity thus complicating the scenario. The dataset has been made available for download[5]. Please note that each set of queries also corresponds to a different class of popularity in the query log. In particular, the first set is made up of queries that are typically in the head of the power-law, while the second and the third set are made up of queries in the torso/tail of the power-law. Furthermore, some of the queries in the third set (containing the longest queries) are singleton queries (i.e., queries that occur only once in the query log).

4.3 Experimental Results

We evaluate S^3 by comparing its performance with those obtained with the original version of the SS algorithm. In particular, for each set of queries in the dataset described above (*short*, *medium* and *long*), we computed average relatedness and average diversity.

[5]Interested readers can download it from: http://hpc.isti.cnr.it/~ceccarelli/doku.php/sss

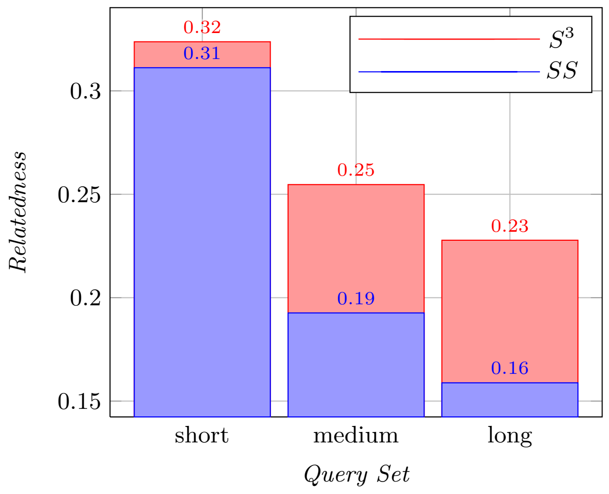

Figure 1: Per-set average relatedness computed between the list of suggestions and the given query.

Figure 1 shows the average relatedness computed for each query q belonging to a particular set of queries. Results confirm the validity of our intuition as, for all the three sets, the results obtained by S^3 are always greater than the results obtained by considering the SS suggestions. It is worth to observe that the entities suggested by S^3 are potentially completely different by the entities annotated in the suggestions of SS. In fact, while in SS we are exploiting only the entities in the titles, in S^3 we are leveraging all the entities in the *whole virtual document*, using the virtual document relevance to boost the most important entities. Furthermore, the longer the queries the more difficult the suggestion of related queries. This happens because long queries occur less frequently in the log and then we have less information to generate the suggestions. If we consider single sets, the highest gain of S^3 in terms of average relatedness is obtained for medium and long queries: this means that relying on entities allows to mitigate the sparsity of user data.

Figure 2 reports the average diversity of the suggestions over the queries of each set. Here, we observe an opposite trend, due to the fact that the longer the queries, the more terms/entities they contain, and the more different the suggestions are. Furthermore, we observe that, for the most frequent queries, SS has a very low performance *w.r.t.* S^3. This happens because in the case of frequent queries SS tends to retrieve popular reformulations of the original query, thus not diversifying the returned suggestions. S^3 does not suffer for this problem since it works with entities thus diversifying naturally the list of suggestions. We leave as future work the study of a strategy for suggesting entities aiming at maximizing the diversity on a list of suggestions.

Let us clarify with an example how the two techniques behave differently. Given the query "*dante*", SS returns the following suggestions: *dante banquet*, *dante boska komedia*, *dante paradiso*, *dante kupferstich*, *dante's divina*, *dante divine comedy*, *dante ali*, *dante alle*. Please note the previously highlighted behavior of SS. The suggestions it produces are often reformulations of the same query, while S^3

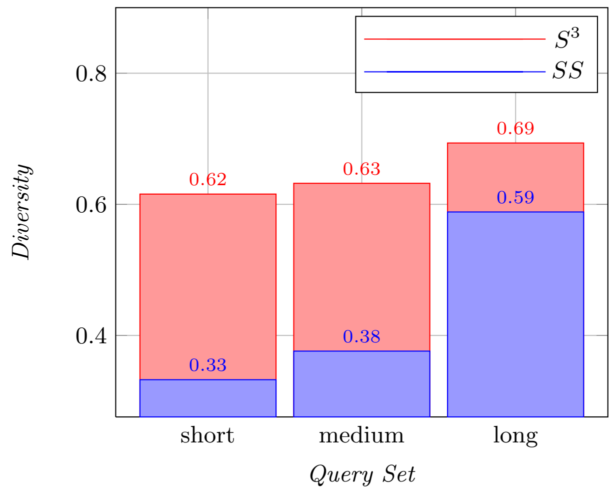

Figure 2: Per-set average diversity computed between the list of suggestions and the given query.

is able to expand the set of suggestions to the entities: *Divine_Comedy*, *Dante_Falconeri*, *Italian_battleship_Dante_Alighieri*, *Inferno_(Dante)*, *Ludovico_Ariosto*, *Sándor_Petőfi*, *Petrarch*, *Dante_Gabriel_Rossetti*, *Convivio* with an average relatedness of 0.48 (SS, 0.43) and a diversity of 0.40 (SS, 0.10).

5. CONCLUSIONS

We explored the use of entities extracted from a query log to enhance query recommendations. We presented our technique and we assessed its performance by using a manually annotated dataset that has been made available for download to favor the repeatability of experiments. The quality of suggestions generated has been measured by means of two novel evaluation metrics that measure semantic relatedness and diversity. This work opens many possible new directions for future research. In particular: i) the study of new measures to define the quality of semantic query suggestions; ii) the refinement of our annotator to manage multilingualism; iii) the enlargement and improvement of the dataset, by also taking into consideration entity attributes.

Acknowledgements We acknowledge the Europeana Foundation for providing us the query logs used in our experimentation. This work was partially supported by the EU project InGeoCLOUDS (no. 297300).

References

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

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

[3] E. Balfe and B. Smyth. Improving web search through collaborative query recommendation. In *Proc. ECAI'04*. IOS Press, 2004.

[4] R. Baraglia, F. Cacheda, V. Carneiro, D. Fernandez, V. Formoso, R. Perego, and F. Silvestri. Search shortcuts: a new approach to the recommendation of queries. In *Proc. RecSys'09*, New York, NY, USA, 2009. ACM.

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

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

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

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

[9] D. Broccolo, L. Marcon, F. M. Nardini, R. Perego, and F. Silvestri. Generating suggestions for queries in the long tail with an inverted index. *IP&M*.

[10] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *Proc. WWW'09*. ACM, 2009.

[11] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 231–238, New York, NY, USA, 2007. ACM.

[12] D. Ceccarelli, S. Gordea, C. Lucchese, F. M. Nardini, and G. Tolomei. Improving europeana search experience using query logs. In *Proc. TPDL'11*, pages 384–395.

[13] D. Downey, S. Dumais, and E. Horvitz. Heads and tails: studies of web search with common and rare queries. In *Proc. SIGIR'07*. ACM, 2007.

[14] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proc. CIKM'10*, pages 1625–1628. ACM, 2010.

[15] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *Proc. SIGIR'11*, pages 765–774. ACM, 2011.

[16] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08*, pages 699–708. ACM, 2008.

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

[18] C. Kang, S. Vadrevu, R. Zhang, R. Zwol, L. Pueyo, N. Torzec, J. He, and Y. Chang. Ranking related entities for web search queries. In *Proceedings of the 20th international conference companion on World wide web*, pages 67–68. ACM, 2011.

[19] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proc. SIGKDD'09*, pages 457–466. ACM, 2009.

[20] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *Proc. WSDM'11*, pages 277–286, New York, NY, USA, 2011. ACM.

[21] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *Proc. CIKM'08*. ACM, 2008.

[22] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. De Rijke. Learning semantic query suggestions. *The Semantic Web-ISWC 2009*, pages 424–440, 2009.

[23] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *Proc. CIKM'07*, pages 233–242, New York, NY, USA, 2007. ACM.

[24] D. Milne and I. Witten. Learning to link with wikipedia. In *Proc. CIKM'08*, pages 509–518. ACM, 2008.

[25] P. Pantel and A. Fuxman. Jigs and lures: associating web queries with structured entities. In *Proc HLT'11*, pages 83–92, Stroudsburg, PA, USA, 2011. ACL.

[26] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proc. KDD'05*. ACM Press, 2005.

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

[28] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proc. WWW'01*. ACM, 2001.