Query Portals: Dynamically Generating Portals for Entity-Oriented Web Queries

Sanjay Agrawal
Microsoft Research
sagrawal@microsoft.com

Kaushik Chakrabarti
Microsoft Research
kaushik@microsoft.com

Surajit Chaudhuri
Microsoft Research
surajitc@microsoft.com

Venkatesh Ganti*
Google
vganti@google.com

Arnd Christian König
Microsoft Research
chrisko@microsoft.com

Dong Xin
Microsoft Research
dongxin@microsoft.com

ABSTRACT

Many web queries seek information about named entities (such as products or people). Web search engines federate such entity-oriented queries to relevant structured databases; the results of those searches are then returned to the user along with web search results. Current federated approaches have two limitations: (i) they often fail to return important results for a broad class of such entity-oriented queries and (ii) the information they return per entity is often inadequate. In this paper, we present the *Query Portals* system that addresses these limitations. The Query Portals system *dynamically* generates a portal for an entity-oriented query. It first provides an overview of the relevant entities and further allows users to drill down to gather additional information on these entities. Our architecture uses a judicious combination of pre-processing and query time techniques so that the query portal can be generated efficiently.

Categories and Subject Descriptors

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

General Terms

Experimentation,Performance

Keywords

Entity search, vertical search, information extraction, portals

1. INTRODUCTION

Many web search queries do not look for web pages per se, but instead are seeking information about named entities like products, people, locations and organizations. A recent study shows that 52.9% of web search queries are entity-oriented queries [11]. For many such queries, the entities of interest reside in a structured database. For example, consider a query such as `[low light digital camera]`. In this case, the user is looking for digital cameras that perform well in low light. This query can be better served by returning names of relevant products from a product database.

The need to search structured databases is illustrated by the proliferation of vertical search engines for products [1], celebrities [2], etc. Current web search engines already federate many web queries to one or more structured databases containing information about named entities like products, people, movies and locations. Each structured database is searched individually and the relevant structured data items are returned to the web search engine. The search engine gathers the structured search results and displays them along side the web search results. However, this approach is not satisfactory due to two key limitations.

• **Silo-ed Search**: Current federated search over each structured database is *"silo-ed"* in that it exclusively uses the information in the specific structured database to find matching entities. That is, it matches the query keywords *only* against the information in the structured database. The results from the structured database search are therefore independent of the results from web search. We refer to this type of structured data search as *silo-ed search* [4].

While the silo-ed search works well for some queries, it would return *incomplete or even empty results* for a broad class of queries [4]. Consider the query `[low light digital camera]` against a product database containing the name, description, and technical specifications for each product. *Canon EOS 30d* may be a relevant product but the query keywords `low light` may not occur in its name, description or technical specifications. Silo-ed search over the above product database would fail to return this relevant product. Although one or more reviews of the product may describe it using those keywords, the structured database may not contain the comprehensive set of reviews of each product necessary to identify the relevant products. Hence, a silo-ed search against the structured database may miss very relevant results [4].

• **Inadequate Information for Drill Down**: When entities in a structured database are found to be relevant for a user's query, current approaches return *only* the information about the entity that is available within the database. The information in the database might be inadequate. Often, the user's need might be better served by information on the web. For example, consider the product *Canon EOS*

*Work done while at Microsoft Research.

SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.

Figure 1: Query Portals screenshot showing relevant entities

30d. The database may have information about the technical specifications and price for this product. However, a user might also be interested in reviews, device drivers, manuals or accessories of the product which are available on the web but not in the database. Providing links to such information would satisfy her information requirement.

1.1 Our contributions

In this paper, we propose the *Query Portals* technology to address the above two limitations by (i) *leveraging web search results* to identify the entities in structured databases that are relevant for an entity-oriented query, and (ii) enabling users to drill down and obtain, in addition to information from the database, specific information *from the web* on these entities. Thus, we create a *portal-like* functionality by providing an overview with a set of entities relevant to a web search query, and then allowing users to drill down further into one or more of these entities. However, unlike a web portal which is static, a query portal is generated *dynamically* for each query.

We now briefly discuss how our approach addresses the two limitations discussed earlier.

• **Going Beyond Silo-ed Search**: Consider our example query `[low light digital camera]`. We observe that although the database may not contain the keywords `low light` in the description of *Canon EOS 30d*, that information is available on the web: several web documents from product review sites, blogs and discussion forums will typically mention *Canon EOS 30d* (and other relevant products) in the context of the query keywords low, light, digital and camera. Therefore, the documents returned by a web search engine for the above query are likely to mention products that are relevant to the user query. Our key insight is that if we can automatically identify the *mentions* of the products in web documents, we can derive the most relevant products by *aggregating* the mentions of products in the top n web search results [6]. Since we complement the information in the database with that available on the web, our approach can return entity results for a much wider range of queries compared to silo-ed search [4].

A screenshot of the set of relevant product entities returned by our Query Portals system for the query `[low light digital camera]` is shown in Figure 1. The shaded region on top shows the query portal output; the traditional search engine results are shown below. The set of relevant entities (grouped by the brand name—Canon, Fuji, Nikon, etc.—in this particular example) provides the user with a quick overview of the product entities related to her query. The green bars represent the relevance of the entities to the user query.

• **Providing Adequate Information for Drill Down**: After looking at the set of all relevant entities, the user may want to obtain more information about one or more specific entities. As mentioned before, the desired information may not be available in the structured database. We address this limitation by pointing users to more information about the specific entity *on the web*. Specifically, we consider two types of information on the web about an entity. First, we suggest *authoritative* web sites where a user can find extensive information about an entity. We identify the authoritative sites at the entity category level first and then percolate it down to the individual entities of the category; we discuss this in details in Section 2.1. Second, we surface *focused web search queries* per entity to enable a user obtain very specific information on the web about an entity. The intuition behind focused queries is best illustrated through an example. Suppose many queries in the query log ask for reviews of digital cameras. We infer that users are interested in reviews for *any* digital camera and surface the focused query `[e reviews]` for any digital camera e. We refer to the union of authoritative web sites and focused web search queries for a specific entity as *entity information links*. We rely on the query logs, click logs, category information about entities, and the snapshot of web documents in order to automatically identify information links per entity.

The entity information links we show for the example entity *Canon EOS 30d* is illustrated in Figure 2. We suggest authoritative web sites such as CNet and comparison shopping sites such as MSN Shopping (now called Bing Shopping). We also suggest focused web search queries (such as `[Canon EOS 30d reviews]`) to enable a user to obtain re-

Figure 2: Query Portals screenshot showing information links for an entity

views, batteries, accessories, drivers, software and manuals for this product. Depending on the user's information requirement, she may choose one or more of these suggestions. Note that *our approach is complementary* to the information about an entity already available in a structured database.

In summary, for entity-oriented queries, the Query Portals system presents, in addition to traditional web search results, an overview of the entities relevant to the query and further enables the user to obtain specific information about any of the entities so surfaced.

2. ARCHITECTURE

We now describe the architecture of the Query Portals system. As shown in Figure 3, the system has pre-processing and query-time processing components.

2.1 Pre-processing Components

The Query Portals system has two pre-processing components: *entity extraction* and *entity information links identification*. We first briefly discuss these two components.

Entity Extraction: The *entity extraction* component takes as input a snapshot of the web documents along with the structured database containing the entities. It outputs a relation consisting, for each URL in the web snapshot, *mentions* of entities in the given database. It analyzes each web document and outputs a list of [url, entityid, position_in_document] tuples. We refer to this relation as the *URL-EntityList* relation. We store this relation in a relational database with an index on the URL column for efficient access at query time.

In principle, our system can build upon any entity extraction technology. We now sketch the approach adopted in the current Query Portals system. For any entity not in the entity database, we cannot provide the rich information to the user for drill down as we have no any additional information about it. Such entities would therefore not be returned as results. We constrain the set of entities that need to be identified in a web document to be from the given entity database. We leverage this entity database membership constraint to develop techniques (i) which can handle a *wide variety of structured data domains*, and (ii) which are also significantly more *efficient* than traditional entity extraction techniques. The task now is to analyze document sub-strings which match with an entity name in the given database [3].

In most realistic scenarios, say for extracting product names, expecting that a sub-string in a web document matches exactly with an entry in a structured database table is very limiting. For example, consider the product entity *Canon EOS 30d*. In many documents, users may just refer to this product by writing *Canon 30d*. Insisting that sub-strings in documents match exactly with entity names in the reference table may lead to failure in extracting these product mentions. Therefore, it is very important to consider *approximate matches* between document sub-strings and entity names in a reference table [9, 5]. Our approach is to generate synonyms for entities, augment the synonyms to the original list of entities and use exact sub-string matching. Thus, we retain the efficiency of exact matching and yet are able to handle approximate matches. The synonym generation technology used in the Query Portals system is described in [7, 8].

Another issue is that of pruning document sub-strings which match an entity name in the database but do not refer to the entity. For example, the distinction between the movie "60 seconds" versus a phrase "60 seconds" (in reference to time) is important while extracting a set of movies. We apply known techniques for this entity recognition step [9, 10].

Entity Information Links Identification: This component identifies the information links for each entity in the given structured database. This consists of (i) a set of authoritative web sites which provide detailed information for the entity and (ii) a set of focused web search queries for obtaining more specific information about the entity. The output of this component is a list of [entityid, information_links] tuples; we refer to it as the *Entity-InformationLinks* relation. We store this relation in a relational DBMS with an index on the entityid column for efficient access at query time.

We now explain how we identify the focused queries automatically. We notice from the query logs that many users are issuing queries of the form `[e X]` where `e` is an entity (string) and `X` is single word or a sequence of few words (e.g., `[Canon EOS 30d reviews]`, `[Canon EOS 30d drivers]`). We refer to `[e X]` as a focused query for the entity e. If users are issuing focuses queries of the form `[e X]` for many entities e of a certain category C, we infer that `[e X]` is an important focused query for *every* entity in the category C. For

example, if users issue focused queries of the form `[e reviews]` for many camera entities e, we identify `[e reviews]` as a focused query for *all* cameras. We also apply other processing (such as stop word removal, stemming, removing approximate duplicates, and a few domain-specific filters) over focused queries to ensure that they are are robust and accurate.

We next consider authoritative sites. The click log shows that for many entities e in a certain category C, users issuing queries of the form `[e]` often click on URLs in the same web site domain D. For example, for camera queries, users may always click on URLs in the domain `CNet.com` or `dpreview.com`. We generate D as an authoritative site for all entities of category C.

In summary, we identify both types of entity information links by *aggregating queries/clicks to the category level* and identifying the dominating focused queries/web site domains. The power of aggregation is key in producing robust entity information links.

2.2 Query-time Components

We now briefly discuss our three query-time components.

Entity Retrieval: The task of entity retrieval is to lookup the URL-EntityList relation (materialized by the Entity Extraction pre-processing component) for each of the URLs in the top n results from a web search engine for the user's query, and retrieve the set of entities mentioned in those documents along with their positions. We use $n = 30$ in our demo as it is a good tradeoff between quality and efficiency. To enable efficient retrieval of entities per URL, we index the URL-EntityList relation on the URL column (see Section 2.1).

Entity Aggregation and Ranking: This component ranks the set of all entities returned by the entity retrieval component and surfaces the top few entities to the user. Our main insights are as follows. First, how and where an entity is mentioned in the individual returned documents provides evidence about its relevance to the query. For example, an entity mentioned closer to the query keywords in the returned document is likely to be more relevant than that mentioned farther away from the query keywords. Second, how often an entity is mentioned among the top web search results also provides an important hint about its relevance to the query. For example, a product mentioned often among top web search results is likely to be more relevant than another product which is mentioned only a few times. Hence, we can identify the most relevant products by first assessing the evidence each individual returned document provides about the products mentioned in it and then *"aggregating"* those evidences across the returned documents. The exact ranking function is described in [4]. We surface only the entities whose scores are above a pre-determined threshold, and rank them in the descending order of their scores.

Information Link Retrieval: The task of this component is to lookup the Entity-InformationLinks relation (materialized by the Entity Information Links Identification preprocessing component) to retrieve the information links for each entity surfaced to the user so that the user can drill down into any of those entities. To enable efficient retrieval of the information links per entity, we index the Entity-InformationLinks relation on the entity id column. We then display the entities by grouping them on entity type (people or products) and an interesting attribute of the entity, say, the brand name as shown in Figure 1. Currently, the attributes we group relevant entities on are predetermined for every entity category.

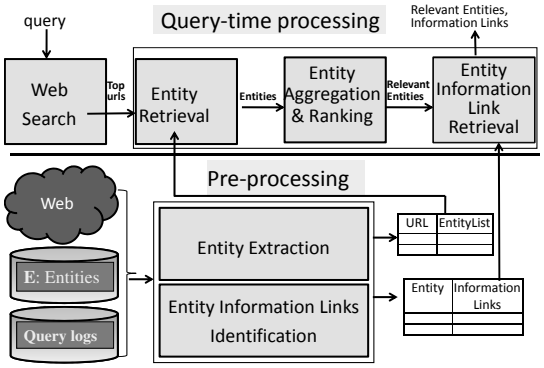

Figure 3: Query Portals Architecture

3. DEMO SCENARIOS

Our insight of complementing the information in the structured database with information available on the web is general and applies to many types of entities. Furthermore, our architecture can handle a wide variety of entity types. In order to emphasize this generality, we will demonstrate the query portal functionality for two entity types, *products* and *people*. First, consider a scenario in which a user is researching cameras that work well in low light conditions. She can ask the query `[low light digital camera]` in our product query portal system and get back the relevant cameras as shown in Figure 1. She can drill down into any of them as shown in Figure 2. Second, consider another scenario where the user is interested in classic romantic movies and looking for actors/directors/producers of such movies (so that she can obtain more information about them, e.g., movies they acted in). She can ask the query `[classic romantic movies]` in our people query portal system and get back the dominant actors/directors/producers etc. of such movies.

4. REFERENCES

[1] http://www.bing.com/shopping/.

[2] http://www.bing.com/xrank.

[3] S. Agrawal, K. Chakrabarti, S. Chaudhuri, and V. Ganti. Scalable ad-hoc entity extraction from text collections. In *VLDB*, 2008.

[4] S. Agrawal, K. Chakrabarti, S. Chaudhuri, V. Ganti, C. König, and D. Xin. Exploiting web search engines to search structured databases. In *WWW Conference*, 2009.

[5] K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin. An efficient filter for approximate membership checking. In *ACM SIGMOD Conference*, 2008.

[6] K. Chakrabarti, V. Ganti, J. Han, and D. Xin. Ranking objects based on relationships:computing top-k over aggregation. In *SIGMOD*, 2006.

[7] S. Chaudhuri, V. Ganti, and D. Xin. Exploiting web search to generate synonyms for entities. In *WWW Conference*, 2009.

[8] S. Chaudhuri, V. Ganti, and D. Xin. Mining document collections to facilitate accurate approximate entity matching. In *Proc. VLDB Endow.*, 2009.

[9] W. Cohen and S. Sarawagi. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *ACM SIGKDD Conference*, 2004.

[10] V. Ganti, A. C. König, and R. Vernica. Entity categorization over large document collections. In *ACM SIGKDD*, 2008.

[11] R. Kumar and A. Tomkins. A characterization of online search behavior. *IEEE Data Eng. Bull.*, 2009.