Toward Query-centric Web Modeling and Crawling

Jiuming Huang [1,2], Haixun Wang [1], Ariel Fuxman [3], Yan Jia [2]

[1]Microsoft Research Asia
[2]National University of Defense Technology, China
[3]Microsoft Research Silicon Valley

ABSTRACT

Crawling and information extraction is perhaps one of the most extensively studied topics in the web age. Albeit much progress has been made in this area, it is still a page-centric, computation intensive process, and more often than not, it relies on manually crafted templates. But how a web site organizes its data has gone through significant changes since the early days of web, and many web sites, especially e-commerce sites, now support *exploratory search*, which presents data to the user through an implicit query interface. In this paper, we conduct a comprehensive survey of exploratory web sites, and we show that traditional page-centric crawling is extremely wasteful and the crawled data is seriously incomplete. We propose a query-centric view of web data, and an automatic crawling framework that crawls web sites by queries instead of by pages. In essence, we make queries first class citizens in modeling web data. This allows us to target the right content in crawling, which not only makes crawling more efficient, but also enables us to collect data that is hidden from page views and hence unreachable to traditional crawling methods. We conduct extensive experiments to demonstrate the advantage of the new model, and the performance of our new crawling method.

1. INTRODUCTION

The web is a valuable source of information. A lot of applications, from search engines to content portals, rely on data extracted from the web. In many cases, a database exists behind a web site, and the goal of crawling is to acquire as much and as complete data from the database as possible. For example, many applications focus on crawling product information on e-commerce sites, and the goal is to obtain as much and as complete information such as the name, brand, price, and other properties of the products [2, 7, 13, 19].

The effectiveness of web crawling largely depends on how web sites present their data. Certainly, web sites present their data in ways to maximize user experience (instead of making the life of a crawler easier). A majority of web sites now organize their data to support so-called *exploratory search* [11], which combines querying and browsing to improve user experience on the web. Specifically, it provides an easy-to-use interface to allow users to construct queries step by step. At each step, the web site presents entities that satisfy the current query to the user. The query can often be succinctly represented in SQL:

```
SELECT * FROM database
WHERE a1 = v1, a2 = v2, ..., ak = vk
```

where `a1=v1, ..., ak=vk` is a list of attribute/value pairs constructed by the user. Many web sites also provide another mechanism, which is keyword search. However, the results of the above queries usually cover entities in the entire database already, which means they cover the results of a keyword search. Thus, as far as the goal of acquiring as much data as possible is concerned, we can focus on SQL-like queries and ignore keyword search.

We surveyed a large number of popular web sites in different categories (Section 2), and the result shows that a large percentage of web sites are *exploratory*, or organized by queries. However, state-of-the-art crawling techniques are still page-centric. When pages are considered as the first class citizen in crawling, and semantics beyond pages, such as queries, is ignored, we may suffer great vulnerabilities such as i) crawling is extremely inefficient and wasteful, and ii) the data crawled is often seriously incomplete.

Crawling is extremely wasteful

Traditional crawling focuses on individual web pages, or on how web sites present their data, instead of on the data itself. In other words, we are crawling the presentation, instead of the data that drives the presentation. When data semantics is ignored, we may waste a lot of efforts on pages that do not contain new information. The number of such pages can be orders of magnitude larger than useful ones.

To see this, consider exploratory sites that allow users to construct queries. A user can first specify she is interested in *handbags*, and then she selects a price range, say *$200 and above*.

```
SELECT * FROM database
WHERE category = 'handbags' AND price >= 200.00
```

Another user might make the selection in the reverse order. That is, she selects price range *$200 and above* first, and then *handbags*. The two queries are exactly the same, however, the URLs that represent the queries are different. Figure 1 shows the URLs of the above example on `Amazon.com`. The URLs are only different but also encoded in a way that is hard to interpret. In other cases, for example, the two `Food.com` URLs (for recipes) in Figure 1 are interpretable, but still, different. In either case, a crawler that does not understand the semantics of the query will consider the URLs as totally different, and end up crawling many redundant pages (all the URLs originate from this page might be prefixed with the current URL).

Consider the two queries in Figure 2 as another example. The two queries differ in the ORDER BY clause, which affects how

VLDB '11, August 29- September 3, 2011, Seattle, WA

$200 & Above → Handbags:
http://www.amazon.com/gp/search/ref=sr_nr_n_5?rh=n%3A10365 92%2Cn%3A!1036682%2Cn%3A1036700%2Cn%3A247493601 1%2Cp_36%3A1253492011%2Cn%3A15743631

Handbags → $200 & Above:
http://www.amazon.com/gp/search/ref=sr_nr_p_36_4?rh=n%3A10 36592%2Cn%3A!1036682%2Cn%3A1036700%2Cn%3A247493 6011%2Cn%3A15743631%2Cp_36%3A1253492011

Rice → Stews:
http://www.food.com/recipe-finder/rice,stews

Stews → Rice:
http://www.food.com/recipe-finder/stews,rice

Figure 1: Different URLs for equivalent queries

```
SELECT * FROM database
WHERE a1 = v1, a2 = v2, ..., ak = vk
ORDER BY PRICE ASCENDING

SELECT * FROM database
WHERE a1 = v1, a2 = v2, ..., ak = vk
ORDER BY PRICE DESCENDING
```

Figure 2: Different outputs for two equivalent queries

the data is returned, instead of what data is returned. However, the pages generated by the two queries are very different. If we cannot identify their equivalence, then we end up crawling all pages (which have unique URLs) that originate from these pages.

Clearly, since there can be a huge number of combinations of attributes, ordering, etc., a crawler that ignores the query semantics will end up crawling orders of magnitude more pages.

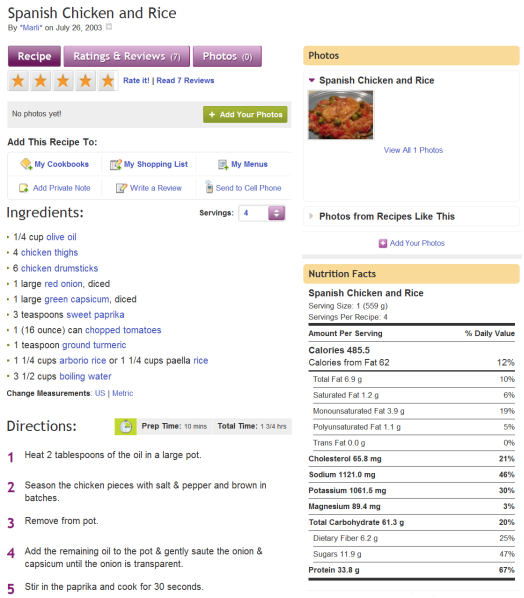


Figure 3: An Entity Page (information about one recipe)

Crawled data is seriously incomplete

Assume an entity has an attribute/value pair $a = v$, for example, *category=woman* or *theme=casual*. In exploratory search, a user may choose $a = v$ when she formulates queries interactively in a step by step fashion. Thus, the information about $a = v$ does not need to appear on the final page that describes an entity, as the user is already aware of it. On the other hand, if the user does

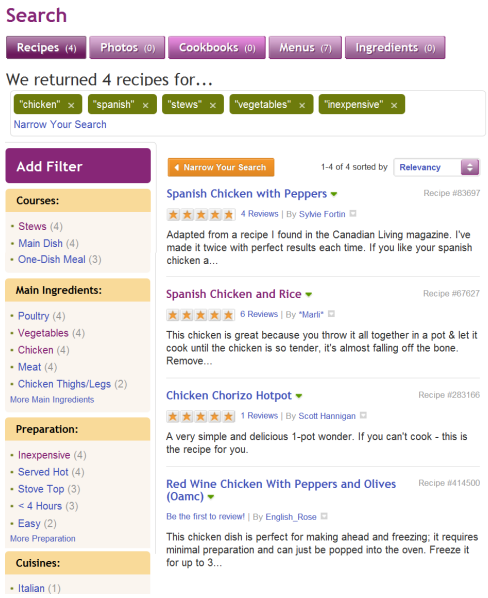


Figure 4: A List Page (links to multiple recipes)

not select $a = v$ in exploratory search, then it might mean the information is not important to the user or the user can derive it from non-text information on the final page (e.g., from the picture of the entity), thus the information does not need to be shown on the entity page either (unless it is essential such as price). As traditional crawling only focuses on the entity pages, it loses information that is embodied in the web exploration process.

This problem is quite universal for exploratory sites across all categories. We use a recipe web site as an example. Figure 3 is an entity page (a page that describes a recipe in this case) on `Food.com` about the recipe "Spanish Chicken and Rice." It contains some detail information about the recipe, including ingredients, cooking directions, and nutrition facts. However, much information, such as the type of *dietary*, the *occasion* the recipe is for, other *major ingredients* it contains, is not shown on the page. But the web site actually contains all such information. In Figure 4, the "Add Filter" section on the left allows users to add searching conditions for recipes, and users can specify types of courses, ingredients, preparation methods, etc. But most of the information that can be specified in filtering are not present on the entity page.

In Table 4 (Appendix), we list all the attribute/value pairs that are used to describe the recipe "Spanish Chicken and Rice." The information is collected from recipe pages and filter links. As shown in the table, more than 67% of the data is not present on the entity page. We evaluated many exploratory sites across different categories, and on average, more than 50% of information about entities is hidden in the dynamic exploratory process. Clearly, data obtained by traditional crawling is seriously incomplete.

Supervised crawling as a naïve solution

The fundamental cause of the above two problems is that the crawler does not know the semantics in the data: It views a web site as consisting of a set of individual pages, and it has no idea that the pages are organized around queries.

To emulate a human interacting with an exploratory web site, the crawler must be able to first formulate a query through the web interface, and then identify results returned by the web site. This is difficult as each step may require the crawler to intelligently follow through many pages.

A direct solution is to manually inject semantics into the crawling process. For example, through the use of manually crafted tem-

plates or through examples created by user interaction, a crawler may learn, for each particular web site, what pages are useful and what part of a page contains content of interest. Apparently, the approach is ad-hoc, and cannot automatically support heterogeneous web sites. More importantly, approaches based on templates or user provided examples are for individual pages only, and they fall short to support exploratory web sites, as on exploratory web sites, much information lies in the dynamic exploration process, instead of in static pages.

Our contributions

We introduced the problem of attribute extraction leveraging browsing structure in a recent paper [9]. In that paper, we presented a proof-of-concept algorithm and successfully applied it to two commercial sites: Amazon and Zappos. We now address the challenges involved in implementing the vision at scale. For example, the algorithm in [9] requires an entire Web site to be crawled before any extraction can be done. We now present an algorithm that performs focused crawling of web sites, thereby drastically improving the efficiency of the system. We also crisply characterize the class of Web sites that can benefit from browsing-aware extraction, and provide exhaustive experimental results on the top commercial sites as per the Hitwise report.

In summary, this paper makes the following contributions: i) We conduct a comprehensive survey of exploratory web sites. We not only show the popularity of exploratory search, but also characterize its benefits and challenges. ii) Based on the query-centric model, we introduce a highly efficient online algorithm for web site crawling. We show that we achieve accuracy and recall of both 90% and higher. iii) We show our approach has great generality, as we can handle over 70% exploratory web sites across different categories. Furthermore, our approach is mostly unsupervised, which means we are not vulnerable to site updates, etc.

2. EXPLORATORY SEARCH: A SURVEY

How web sites organize and present data has evolved significantly since the early days of the Web. In this section, we introduce the concept of exploratory search and browsing, and survey various web sites to show the characteristics of exploratory web sites.

Exploratory search [11] combines querying and browsing in web surfing. It provides hierarchical or multidimensional browsing options (e.g., the left panel on the page shown in Figure 4) through which users can refine their search. This is paritcularly helpful to users who do not have a very specific goal in browsing (e.g., a customer who wants to buy a handbag but has not decided on the brand, price range, color, size, etc.)

The exploratory search model has been widely adopted by web sites across different domains ranging from e-commerce to digital library. We conducted a comprehensive survey over 443 highly ranked[1] web sites in 20 major domains (e.g., shopping, travel, real estatate, etc). Figure 5 shows that 127 out of the 433 web sites (28.7%) use exploratory search. In particular, shopping web sites are the most aggressive in adopting the exploratory model: 97 out of top 133 e-commerce web sites (72.9%) use exploratory search.

As we mentioned in Section 1, a crawler that ignores the exploratory search semantics and focuses on the final pages of each entity will miss a lot of information about the entity. Our survey showed that among the 127 web sites that use exploratory search, only 29 or 23% put complete information about each entity on the entity page. A large majority (77%) contains incomplete information. Note that on some web sites, e.g., `Ebags.com`, the entity

[1]The ranking is obtained from `Hitwise.com`, a leading online competitive intelligence service.

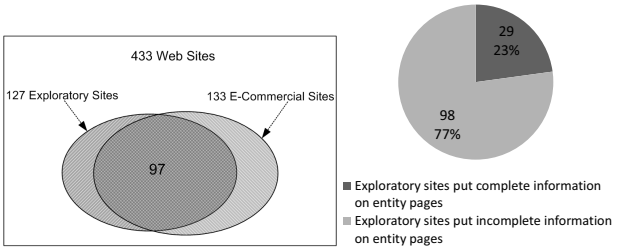


Figure 5: Survey of 433 highly ranked web sites.

page may show a path (e.g., Handbags → Girls → $200 Above) that indicates how the user reaches the entity via exploratory search. This can be used to make up for the information lost in browsing. However, such information is not complete either (e.g., there are multiple paths that can lead to the same product). Our survey finds that only 25% of exploratory sites show multiple paths on the entity page, and close to 30% sites do not provide any such information.

In this paper, we introduce a query-centric model for exploratory web sites, and we develop a crawler based on the model. As we show in experiments, our approach is general as it can handle 107 out of the 127 (84.3%) exploratory web sites. Please see Table 1 in Section 5, and Tables 5 and 6 in Appendix D for details.

3. QUERY-CENTRIC MODELING

Exploration Queries

An exploratory web site is a web site that allows the user to browse for entities of interest. The browsing is typically performed by following links that return *result pages* or *list pages*. The list pages contain listings of entities, with links to pages that provide details about the actual entities (*entity pages*).

Let E be the set of entities that an exploratory web site provides information for. Assume that each entity can be described by a set of attributes $A = \{a_1, \ldots, a_m\}$. Conceptually, an exploratory web site executes *exploration queries* on the set of entities E using some of the attributes in A. To be more precise, these exploration queries return a subset of E that satisfies some conjunctive expression over that set of attributes in A.

To understand the exploration queries in relational terms, assume that each entity in E has a unique identifier. Let T be a table with schema $\langle id, a_1, \ldots, a_m \rangle$. Then, an exploration query is of the form:

$$q: \quad \textbf{SELECT id FROM T WHERE } a_1 = v_1, \ldots, a_k = v_k$$

where $a_1 \ldots a_k$ are attributes in A and $v_1 \ldots v_k$ are some constants (attribute values). Without loss of generality, we convert predicates that contain numerical comparisons to predicates without comparisons. This is done by introducing new attributes. For example, given a predicate $a_i > 100$, we convert it into a new predicate $a_{i>100} = \texttt{True}$ where $a_{i>100}$ is a new attribute. Thus, we only need to handle predicates in the form of $a_i = v_i$.

We will represent the queries by just using the conjunctive part of the WHERE clause as follows: $q = \{a_1 = v_1, ..., a_k = v_k\}$; and we will denote as E_q the set of entity (ids) returned by q. We will employ the standard database-theoretic notion of query inclusion, where $q \subseteq q'$ if $E_q \subseteq E'_q$. Given a query q, We are particularly interested in its drill-down query q', which contain one more predicate that q, that is, $q' = q \cup \{a_{k+1} = v_{k+1}\}$. In our approach, we rely on the drill-down relation to eventually recover queries on an exploratory web site.

Query-centric modeling of an exploratory web site

Just like any web site, an exploratory web site consists of pages P and links L to pages. But more than that, an exploratory web site describes a set of entities E, and supports a set of exploration queries Q. Specifically, each entity is described by an *entity page*. Search and browsing is performed by following *query links* that return *list pages*, where each query link represents a query in Q, and each list page contains listings of entities that satisfy the query. More precisely, we model an exploratory web site as follows:

DEFINITION 3.1. *(Query-centric Modeling) An exploratory web site is a tuple (P, L, E, Q, f) such that:*

- *P is the set of pages, including entity pages, list pages, and other pages;*
- *L is the set of links, including query links and other links;*
- *E is the set of entities described by the exploratory web site;*
- *Q is the set of queries supported on the exploratory web site;*
- *$f : Q \rightarrow 2^P$ is a function that maps a query to a set of list pages;*

The goal of *exploratory-based information extraction* is to associate entities to their attributes via the queries that can be formulated. However, the problem lies on the fact that the queries are not explicit in the exploratory web site: all that the crawler sees is a set of pages, each page containing a set of links. Thus, the *exploratory-based information extraction problem* is to first i) recover the queries Q that can be formulated on the web site, and then ii) for each $q \in Q$, find $E_q \in E$ that satisfy q. Once the problem is solved, then given q and E_q, we can associate entities in E_q with attribute values defined by q.

Properties of exploratory web sites

The model suggests a new way of thinking about a web site. A web site is more than a set of web pages: It contains structures that support a question-and-answer purpose. However, the model itself does not tell us directly how to derive structures and semantics out of a set of pages.

Nevertheless, the logic of question-and-answer leads to the following assumptions, which enable us to derive the hidden structures and semantics. Before we go to the details, we introduce some notations. Let P_q be the set of list pages returned by $f(q)$:

$$f(q) = P_q \tag{1}$$

The list pages P_q contains links to entities E_q that satisfy the query q. However, the links to E_q are mixed with other content on P_q. We abuse notation and write

$$P_q \supset E_q \cup Q_q \cup L_q \tag{2}$$

to indicate that P_q contains the following content: E_q, which represents links to entities that satisfy q; Q_q, which represents links to queries that are related to q; and L_q, which represents other links.

Assumption 1.

$$P_q \cap P_{q'} = \emptyset \text{ if } q \neq q' \tag{3}$$

The assumption is that different queries do not share list pages, even for queries that return the same set of entities. In other words, each list page can only "belong" to one query. In theory, for a page $p \in P_q$, it not only contains entities that satisfy q, but also links that are related to q. Thus, if $q \neq q'$, then $Q_q \neq Q_{q'}$, which means pages belonging to different queries have different content, even if the query results are the same. In practice, the pages are generated by an algorithm using some template and q as input, then for different inputs, the pages are different.

Assumption 2.

$$E_{q'} \subseteq E_q \text{ if } q' \supset q \tag{4}$$

This assumption is quite straightforward: a more selective query has a smaller set of results. This assumption is used to identify relationships among queries, and then reveal the content of the queries.

Assumption 3.

$$\text{if } E_{q'} \subseteq E_q \text{ and } q' \in Q_q \text{ then it is likely that } q' = q \cup \{a_i = v_i\} \tag{5}$$

If $E_{q'} \subseteq E_q$, then there is a possibility that $q' \supset q$, although q' and q can be totally irrelevant. But we know from $q' \in Q_q$ that the two queries are related. In an exploratory web site, users are allowed to construct queries by adding/removing attributes one by one. We can hence classify queries in Q_q into 3 groups: i) queries that contain one more attribute than q; ii) queries that contain one less attribute than q; and iii) other queries (e.g., queries for shoes and accessories when the user is browsing outfits). Thus, a very likely scenario is the first case, that is, $q' = q \cup \{a_i = v_i\}$.

Note that Assumption 3 does not always hold. For example, in Q_q, there might exist a query $q'' = q \cup \{a_i = v_i, a_{i+1} = v_{i+1}\}$, and certainly we have $E_{q''} \subseteq Q_q$. This creates some confusion. However, our algorithm will also detect the relationship between q'' and q', and the relationship between q' and q, using the same mechanism, which help eliminate the confusion.

4. SEMANTIC CRAWLING

Given an exploratory web site, our goal is to find every q and its corresponding E_q. As q is essentially a set of attribute/value pairs, $\{a_i = v_1, \cdots, a_k = v_k\}$, we know that each entity in E_q has those attribute values. Besides, the method we use must be scalable and general.

4.1 Entity Pages and List Pages

A fundamental challenge we are facing is how to discover queries from pages and links. We differentiate between two types of pages:

- An *entity page* describes a single entity. On most exploratory sites, it is easy to identify entity pages. For example, entity pages on e-commerce sites usually contain an "Add to Cart" (or its equivalent) link.
- A *list page* contains one or more links to entity pages, and it must not be an entity page itself. Note that an entity page may also contain links to other entities (e.g., related products), but that does not make it a list page.

On Amazon.com, for example, each book has its own page (entity page), and the best-sellers page (list page) contains links to best seller books. Using the above definition, once entity pages are identified, we can identify list pages in a straightforward way.

4.2 An Offline Approach

In this section, we describe an algorithm that allows us to recover all queries and their associations to entities. The method crawls the entire exploratory site first, and mines queries from the crawled data.

Based on Assumption 1 (Eq 3), which says a list page belongs to at most one query, we can cluster list pages so that each cluster

correspond to (at most) one query. To do this, we need to measure the similarity between two list pages. Eq 2 says that in addition to entity links, list pages also contain L_q and Q_q. The set of pages in P_q are usually generated by a same mechanism based on a single template. Thus, although each $p \in P_q$ contains different entity links, the L_q and Q_q links are roughly the same, as they are generated by the same template and the same q as input. Given a list page x, we collect all the links on x. We filter out all entity links (including E_q and some links in L_q that point to entities), and denote the remaining links as l_x. We know that l_x contain Q_q and part of L_q. Then, for two list pages x and y, we use Jaccard similarity coefficient $\frac{|l_x \cap l_y|}{|l_x \cup l_y|}$ to measure their similarity. Based on the similarity, we cluster list pages by queries.

From the resulting clusters, how do we reveal the queries they represent? Let P_q and $P_{q'}$ denote two clusters corresponding to two queries q and q' respectively. Although the queries are "unknown," that is, we do not know the attribute/value pairs corresponding to query q and q', it turns out that we may be able to decide if q and q' has the relationship of $q' = q \cup \{a_i = v_i\}$. We do this by taking the following steps.

1. From P_q and $P_{q'}$ we find E_q and $E_{q'}$. This is not always trivial, because not every entity link in P_q belongs to E_q (e.g., a list page on an e-commerce site contains links to productions under promotion.) However, we can use additional clues, e.g., entity links in Q_q are often inside the same DOM structure on the list page.

2. We try to invoke Assumption 3. That is, we check whether $E_{q'} \subseteq E_q$ and $q' \in Q_q$ are true. For the same reason as in the previous step, we relax the condition $E_{q'} \subseteq E_q$ by checking if $\frac{|E_{q'} \cap E_q|}{|E_{q'}|}$ is close to 1.

3. If Assumption 3 holds, we decide what is the attribute/value pair $\{a_i = v_i\}$. Assume the following snippet contains the link to q':

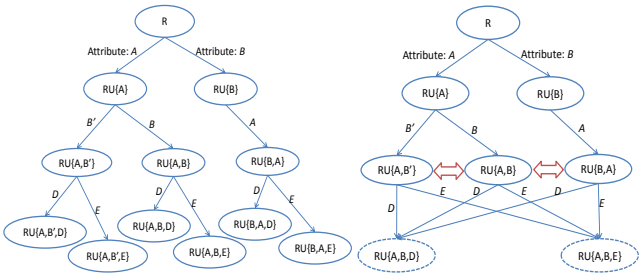
```
<h2>Brand</h2>
...
<a href=http://link-to-query-q'>Nike</a>
...
```

   From the anchor text, we know q' is more specific than q by having an additional descriptor "Nike." We conclude that $q' = q \cup \{\text{brand} = \text{Nike}\}$, as either we already know "Nike" is a brand or we find the name of the attribute "Brand" in the DOM structure one level above the link.

After we discover all the relationships, we obtain a directed graph, where each node represents a query, and each edge represents a relationship between two queries. Specifically, an edge $q_{i-1} \rightarrow q_i$ means q_i is more specific than q_{i-1} by having an additional $a_i = v_i$ condition. Let $q_0 \rightarrow \cdots \rightarrow q_k$ be a longest path in the graph (there is no incoming edges to q_0 and no outgoing edges from q_k). Then, we know $q_i = \{a_j = v_j | j = 1, ..., i\}$. Thus, we recover all the queries in the system.

4.3 An Online Approach

In Section 4.2, We crawl the entire web site, then we recover all the queries from the crawled data. This is however extremely wasteful, as we may crawl orders of magnitude more pages than necessary. To see this, assume we are crawling an e-commerce web site. Consider shoes of size 38, of red color, for ladies. We can reach products in this category by exploring the site along the path of *Shoes* $\rightarrow$ *Red* $\rightarrow$ *Size 38* $\rightarrow$ *lady*. But there are many other paths to reach these products, and in the worst case, we have $N!$ paths, where N is the number of attributes. Furthermore, products in a category can be presented in many different ways, for example, they can be sorted by price, popularity, or other features. Note that each combination of features as well as each presentation corresponds to pages with unique URLs. A crawler that does not understand the semantics of the underlying data will treat each URL as a new source of information, and explore all the structures from that URL.



(a) as a Query Tree (b) as a Query DAG

Figure 6: Organization of an Exploratory Web Site

In this section, we introduce a method that avoids unnecessary crawling. As we crawl an exploratory web site, we discover queries and the relationship among queries in an online manner. The queries we have discovered form a set of trees, or a forest. Figure 6(a) shows one tree in the forest. Each tree node represents a query. Query R is represented by the root node because we have not yet discovered any query that is more general to R. Query represented by each node is denoted in the form of $R \cup \{A, B\}$, where A and B are attributes. We may have multiple nodes correspond to the same set of attributes, for example $R \cup \{A, B\}$ and $R \cup \{A, B'\}$, which is probably because how the data is presented is different (e.g., in one set of pages, entities are sorted by attribute B).

Although we do not know the content of R (the attribute/value pairs it contains) since we are still in the middle of crawling, we can still prune unnecessary crwaling. Note that in Figure 6(a), we know that the nodes labeled $R \cup \{A, B\}$, $R \cup \{B, A\}$, and $R \cup \{A, B'\}$ represent three equivalent queries, although R is still unknown to us. However, since the sets of pages $P_{R \cup \{A,B\}}$, $P_{R \cup \{B,A\}}$, and $P_{R \cup \{A,B'\}}$ are unique, traditional crawling will explore the entire tree below the three nodes as shown in Figure 6(a), which is totally wasteful. In our approach, we only crawl the subtree under one query in an equivalent set of queries. This converts the tree into a DAG as shown in Figure 6(b), through which we reduce the crawling space dramatically.

The challenges lie in online detection of equivalent queries. We accomplish this task using two procedures:

Online Clustering. Our goal is the following: Starting from a list page x, which may belong to some unknown query, find all the entities that satisfy the unknown query. This is done through clustering, using the principle outlined by Assumption 1. However, since we perform clustering in an online manner, we must first find pages that are potentially part of the cluster.

The intuition is that list pages that belong to the same query form a connected graph, e.g., through pagination links that connect them. In other words, starting from any of them, we should be able to find all the other list pages by following the links. Figure 7 illustrates the idea. Starting from list page x, we crawl all the hyper links on x. Then, we identify those hyper links that point to list pages, and we denote such links as l_x. Assume one of the link in l_x points

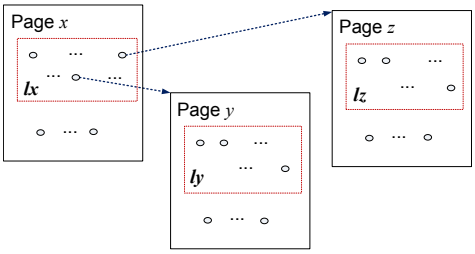


Figure 7: Online Clustering

to page y. For each such y, we use the same method to find l_y, that is, hyperlinks on page y that point to list pages. Then, based on Assumption 1, x and y should be list pages that belong to the same query if l_x and l_y are highly similar (using Jaccard similarity coefficient as described in Section 4.2). We do this recursively, and we stop at page x if we cannot find any y such that l_x and l_y are highly similar. Finally, we obtain a cluster of pages that correspond to the same query. The entire process is outlined in Algorithm 1 (see Appendix).

Online Query Discovery. Our goal is to derive a query's content (its attribute/value list) from the (drill-down) relationships in a query DAG. The challenge is that, unlike what we have described in Section 4.2 where we crawl the entire web site and build the entire query graph before we discover queries from the query graph, we must discover queries from DAGs which are still being constructed.

Each time we call `OnlineClustering` (see Algorithm 1), we get an unknown query R, which can be considered as a root node of a new query DAG, as the one shown in Figure 6(b). As we repeatedly invoke `OnlineClustering`, we either expand a current DAG (if we can find the relationship between the new query and an exiting query in an exiting DAG), or create a new DAG. In our approach, we maintain a forest of DAGs, and the online query discovery process is a process of growing DAGs, creating new DAGs, or merging multiple DAGs, until no more changes can be made.

The central question is how to establish relationships between two nodes that represent two queries. We are interested in two relationships: i) Query R is more specific than R' by having one more attribute/value pair. If we detect this relationship, we connect the two nodes by an edge and label the edge with the attribute/value pair, as edges in Figure 6(b). ii) Query R and R' are equivalent. If we detect this relationship, we merge the two nodes (Figure 6(b) depicts the equivalency by using double arrows for presentation purpose).

We use the same method as in Section 4.2 to establish the first relationship between two queries (information about the two queries are derived by the `OnlineClustering` procedure). To establish an equivalent relationship, we check two conditions: a) The entities that satsify the two queries must be the same. This is a necessary condition but not a sufficient condition, as two different queries can have same results. b) The set of outgoing edges (attributes) from the two nodes must be the same. This is under the assumption that each attribute is used once in the query. If both conditions are satisfied, we consider these two nodes are equivalent.

We then maintain a data structure F, which contains a set of DAGs, or a forest of DAGs. For new list pages we gather on the web site (a good starting point is the root page of the query hierarchy), we invoke `OnlineClustering` to discover the query that the list page belong to, as well as immediate child queries below this query. The result is then added into F, which may be used to expand an exiting DAG, merge two DAGs, or create a new DAG in F. The entire process of online query discovery is outlined in Algorithm 2 (see Appendix).

5. EXPERIMENTS

Generality of Query-Centric Modeling. In Section 2, we surveyed 433 highly ranked web sites and found 127 of them are exploratory web sites. We tested all of them, and found our query-centric model and our information extraction method based on the model are general. Table 1 shows we can successfully handle 107 out of 127 (84.3%) exploratory web sites. There are some special cases. Some web sites require users to enter keywords (e.g., name of a state) instead of letting users choose from a list. However, if we obtain the list of keywords in advance, our approach can still handle such web sites. Still, there are 20 sites we cannot handle, and the reason is they employ complex user interface. For example, `Zagat.com` allows multiple selections to take effect asynchronously using Ajax. Another web site, `Compuplus.com` uses HTML FORMs and requires users to press the "submit" button to refine the search. To support such web sites, we need further customization of the crawler.

Domain	Can be handled	Need keyword input	Cannot be handled
Shopping	52	0	11
Travel	12	0	2
Real Estate	10	0	1
Education	8	0	0
Food	4	4	0
Restaurant Search	4	0	5
Entertainment	2	3	1
Yellow Page	1	2	0
Health	1	0	0
Lifestyle	1	0	0
Digital Library	0	3	0
Total	**95**	**12**	**20**

Table 1: The generality of our approach

Datasets for Performance Study. We select 6 web sites from the 107 exploratory sites for more experiments (precision, recall, overhead, etc). Table 2 lists some basic information about the 6 web sites. For convenience, we denote them as $D1$ to $D6$.

We choose these 6 web sites to maximize diversity and to enable large-scale performance study. The 6 sites are from different domains including online shopping, real estimate, food and restaurants. Furthermore, $D1$ (`Homes.com`) does not contain any query information on its list pages or entity pages, so no template-based (Wrap) approach can extract the hidden attributes from it. We use $D1$ to demonstrate our ability of discovering hidden attributes. $D2$ (`Food.com`) is unique because queries are embedded in URLs without encoding (as shown in the example in Section 1). This gives us a nice "labeled" dataset for evaluating the precision of our algorithm. In $D3$ (`menupage.com`), the complete information of an entity is contained in each entity page. Thus, it can be extracted easily by using traditional technologies. We use $D3$ to evaluate the correctness of our approach. $D4$, $D5$ and $D6$ are online shopping sites (which are the majority of existing exploratory sites). They have a large number of entities and web pages. In particular, $D4$ contains almost all of the typical problems of exploratory sites. We use it to compare the overheads of online approach and offline approach.

Besides our algorithm, there are four ways to obtain attributes

	Web site	Content type	Subcategory	Method	# of real entities	# of real attributes	# of entity-attributes
D1	homes.com	Real Estate	Subset of New York	Manual label	32	34	336
D2	food.com	Food	Entire Site	URL Extraction	434,854	476	5,542,792
D3	menupage.com	Restaurant	American New	Entity page	560	87	57,475
D4	amazon.com	Shopping	Shoes	Templates	248,995	156	1,111,831
D5	shopping.yahoo.com	Shopping	Electronics	Templates	115,866	1,165	670,034
D6	taobao.com	Shopping	Cell Phone	Templates	217,945	127	1,998,446

Table 2: Datasets

Dataset	# of uncovered entities	# of uncovered attributes	# of uncovered entity-attributes	# of page scans	# of crawled pages	Precision	Recall
D1	32	34	336	10,121	6426	1.0	1.0
D2	434,854	476	5,542,792	5,124,627	4,702,653	1.0	1.0
D3	560	87	57,475	15,513	11,120	1.0	1.0
D4	248,995	156	1,077,444	2,183,503	1,150,020	0.97	0.94
D5	115,748	1,165	649,522	2,567,322	1,791,050	0.98	0.95
D6	217,933	127	2,107,057	2,348,836	1,335,548	0.92	0.97

Table 3: Final Results

and values associated with entities from the above datasets: manual labeling, automatic extraction from URL, extraction all the information from entity pages, and the template-based naive solution discussed in Section 1. First, we derive $D1$ by manually labeling data in a subcategory of `Homes.com`. Since manual labeling is costly, we choose a small subcategory (derived from query {city=New York, bath rooms=3, neighbor=yorkville}) with only 336 entity-attribute pairs. Second, if attributes and values are interpretable from URLs, we can get high quality real data. Of course, this is not the case for most of the web sites sites except `Food.com`. So we use the entire `Food.com` data as our second dataset $D2$. Third, we use regular expression to extract data from entity pages on `menupage.com`. Finally, the naive solution described in Section 1 demands complicate templates. So we only design templates for the shoes department of `Amazon.com`, the electronics department of `shopping.yahoo.com` and the "cell phone" data of `taobao.com`.

Evaluation Metrics. We use precision and recall to evaluate our method. We focus on attributes and values that are embedded (hidden) in the browsing process rather than those on the entity pages. Given a set of web pages, we define precision and recall as:

$$Precision(S) = \frac{|R(S) \cap M(S)|}{|M(S)|} \tag{6}$$

$$Recall(S) = \frac{|R(S) \cap M(S)|}{|R(S)|} \tag{7}$$

where S is a dataset (a web site); $R(S)$ is the set of hidden entity-attribute pairs; $M(S)$ is the set of entity-attribute pairs extracted from S by our approach.

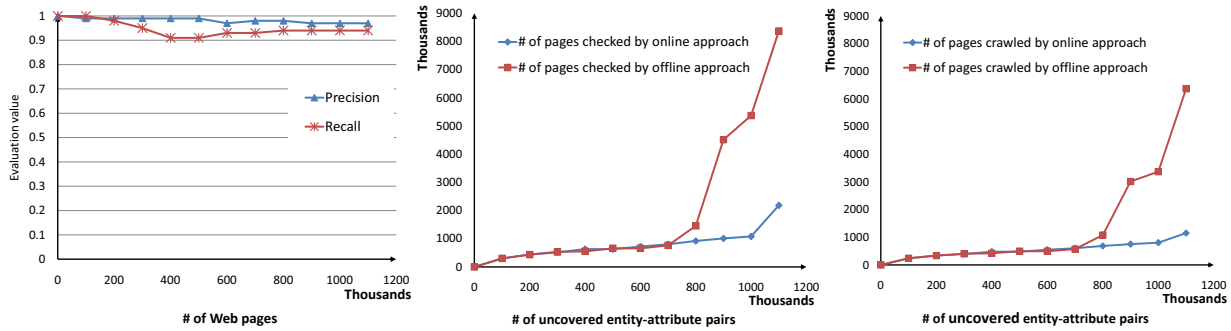
We also evaluate the overhead of our approach. The overhead includes the crawling (network transfer) overhead, and the computation overhead. Crawling (network transfer) overhead is measured by the number of pages we download from the web. Our algorithm may scan each downloaded page multiple times to extract links from the page, and each scan has linear cost. Therefore, for computation overhead, we use the number of page scans as the measure.

Experimental Results and Discussion. Table 3 shows the results for the 6 web sites. For each dataset, we list the precision, recall, as well as the number of discovered entities, attributes, entity-attribute pairs, scanned pages and crawled pages. Besides the results in Table 3, we also study $D4$ (`Amazon.com`) in more depth, and we show the results in Figure 8.

Specifically, as shown in Table 3, we achieved 1.0 precision and recall on $D1$, which shows our approach discovers hidden attributes well. Of course, $D1$ is small. Our performance on larger datasets, including $D2$, $D4$, $D5$ and $D6$, shows our approach is quite stable in achieving high precision and recall. Moreover, the fact that these web sites come from various domains also proves the generality of our approach. Furthermore, the number of scanned pages and crawled pages are in linear proposition to the number of entity-attributes, which shows the efficiency of our approach.

The precision and recall are affected by two factors: i) the content on the list pages, and ii) the frequency of updates on the web site. List pages may contain entities that do not satisfy the query the list pages belong to. For instance, list pages on `Amazon.com` may contain products under promotion. This is the reason why precision on datasets $D4$, $D5$ and $D6$ did not reach 100%. Whereas for $D1$, $D2$, $D3$, the precision is 100% because the list pages of these web sites do not contain unrelated entities or the unrelated entities can be easily filtered. The frequency with which a web site updates its content affects the recall of our method. Our method detects the relationship between two queries by checking the set inclusion relationship between the set of entities that satisfy the queries. Let x and y be two sets of entities. In each snapshot, $y \subseteq x$ holds. However, crawling takes time, and updates may take place during the crawling of x and y, which may lead to $y \not\subseteq x$. This prevents us from discovering the relationship between the queries that x and y correspond to, and hence we may not be able to discover some entity-attributes. This is why recall cannot reach 100%. Obviously, shopping sites update more frequently than recipe sites and restaurant sites. So the recalls of $D1$, $D2$ and $D3$ reach 100%, while the recalls of $D4$, $D5$, and $D6$ are lower.

The precision and recall of our approach are stable. Figure 8(a) shows how the precision and recall change when the page set grows (for $D4$ `Amazon.com`). As the dataset becomes larger, the recall decreases a bit and then recovers and keeps at about 0.94. The reason is the same as discussed above. Amazon.com update its



(a) Precision and recall (b) Computing overhead (c) Crawling overhead

Figure 8: The Performance

products quite frequently. When the dataset becomes large, some crawled pages become outdated due to the large time span in crawling. The reason why the recall goes up again is because the queries we are crawling becomes more and more specific (containing more and more predicates). As a result, the results of those queries become smaller. The time span of crawling a smaller set of results is shorter. So the effect of the web site updates becomes less significant. On the other hand, the precision also goes down a bit and then goes up and maintain at a high level as the dataset grows. The reason is that the number of unrelated entities (e.g., products under promotion) on list pages is finite.

From Table 3, we can see the overhead of $D1$, $D2$ and $D3$ are lower than others. This phenomenon is caused by the page layout of the three web sites. Every list page on these three web sites contains all of the attributes as its query refinement options. Therefore, given a list page, say x, after uncovering the queries associated by the refinement links on x, our approach can obtain all entity-attribute pairs, as they can be retrieved by the query tree (see Figure 6(a)) of x. Thus, the approach can discover all entity-attribute pairs quite fast.

Finally, the online approach is far more efficient than the offline approach. Figure 8(b) and 8(c) show the overhead comparison of the online and the offline approach. Obviously, to discover more entity-attribute pairs, the pages need to be scanned or crawled are growing. To discover the last 10% part of pairs, the number of pages which online approach should scans or crawls grows at a bit higher speed than before because most of the queries uncovered from this part of pages are pruned and the pruning certainly will take some overhead. On the other hand, when 50% entity-pairs has been discovered, the overhead of the offline approach ascends rapidly. As mentioned, the total number of list pages possibly is geometric multiple of the attributes. That is why the offline approach is costly. Due to the high cost, the offline approach is impractical, especially for sites which contain huge mass of data.

6. CONCLUSION

How to effectively handle heterogenous information sources is a core problem to information extraction from the web. Most web crawling and information extraction approaches rely on manually crafted templates, and the process is page-centric, that is, it focuses on individual pages that contain information of interest. We show that a large category of web sites, especially e-commerce web sites, organize their data to support "exploratory search." In the new scenario, web browsing becomes an interactive, query-and-answer process. A crawler that does not understand the process will have intrinsic vulnerabilities. We propose a query-centric approach, which automatically injects the "exploratory search" semantics into crawling and information extraction. Our results show that the approach is general (70% of exploratory sites can be handled automatically), efficient (reducing crawling cost by orders of magnitude), and has high quality (precision and recall of 95% and beyond).

7. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD*, page 337348, 2003.

[2] D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the world wide web. In *ICDCS*, 2002.

[3] M. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. In *PVLDB*, volume 1, 2008.

[4] S. Chakrabarti. *Encyclopedia of Database Systems*, chapter Focused Web Crawling, pages 1147–1155. 2009.

[5] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, pages 109–118, 2001.

[6] N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: An approach based on a probabilistic tree-edit model. In *SIGMOD*, 2009.

[7] R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1), 2006.

[8] R. Gupta and S. Sarawagi. Answering table augmentation queries from unstructured lists on the web. In *PVLDB*, volume 2, 2009.

[9] J. Huang, H. Wang, Y. Jia, and A. Fuxman. Link-based hidden attribute discovery for objects on web. To appear at EDBT, 2011.

[10] K. Lerman, L. Getoor, S. Minton, and C. Knoblock. Using the structure of web sites for automatic segmentation of tables. In *SIGMOD*, 2004.

[11] G. Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 49(4), 2006.

[12] H. Nguyen, T. H. Nguyen, and J. Freire. Learning to extract form labels. In *PVLDB*, volume 1, pages 684–694, 2008.

[13] K. Probst, R. Ghani, M. Krema, A. Fano, and Y. Liu. Extracting and using Attribute-Value pairs from product descriptions on the web. *From Web to Social Web: Discovering and Deploying User and Content Profiles*, 2007.

[14] S. Sarawagi. Information extraction. In *Foundations and Trends in Databases*, volume 1, pages 261–377, 2008.

[15] F. Wu and D. Weld. Autonomously semantifying wikipedia. In *CIKM*, 2007.

[16] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW*, 2005.

[17] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. 2d conditional random fields for web information extraction. In *ICML*, 2005.

[18] J. Zhu, Z. Nie, J. Wen, B. Zhang, and W. Ma. Simultaneous record detection and attribute labeling in web data extraction. In *KDD*, 2006.

[19] J. Zhu, Z. Nie, J. R. Wen, B. Zhang, and W. Y. Ma. Simultaneous record detection and attribute labeling in web data extraction. 2006.

APPENDIX

A. RELATED WORK

Much work has been done on the problem of extracting entity attribute information from Web pages. Some techniques assume the existence of an underlying template in Web sites and are based on wrapper induction (e.g., [5, 1, 6, 16]); while others are template-independent and rely on machine learning models (e.g., [10, 17, 18, 15]; see [14] for a survey). Another line of work exploits lists and tables on the Web for attribute extraction [3, 8]. Some of these approaches leverage, like us, the interrelation between detail and listing pages [10, 18]. Despite their differences, all these techniques have something in common: they assume that the attribute information appears directly associated to the entity (either on the entity detail page or as a record in a listing page). That is, none of them exploit the browsing structure of Web sites, which is the main contribution of our paper.

We introduced the problem of attribute extraction leveraging browsing structure in a recent paper [9]. In that paper, we presented a proof-of-concept algorithm and successfully applied it to two commercial sites: Amazon and Zappos. We now address the challenges involved in implementing the vision at scale. For example, the algorithm in [9] requires an entire Web site to be crawled before any extraction can be done. We now present an algorithm that performs focused crawling of web sites, thereby drastically improving the efficiency of the system. We also crisply characterize the class of Web sites that can benefit from browsing-aware extraction, and provide exhaustive experimental results on the top commercial sites as per the Hitwise report.

Our system performs a focused crawl of Web pages in order to obtain the relevant entity attribute information. In contrast, in the literature on focused crawling (see [4] for a survey), the goal is to gather pages in a category (not entity attributes). This is a fundamental difference, because our crawl is "focused" not by a page classifier (as in traditional focused crawling), but by the "queries" that we associate to the browsing elements of the Web site.

In our work, we discover links that contain attribute information (drill-down links). The attribute information may also be available in form elements (e.g., drop-down boxes). In such cases, it would be possible to leverage existing work on labeling Web form elements (e.g., [12]). Notice that such work is orthogonal to ours, as it does not deal with the problem of associating entities with structured information.

B. CRAWLED DATA IS INCOMPLETE

Table 4 uses food.com to demonstrate that a naive crawler may miss a lot of important information about entities of interest if it ignores the semantics of exploratory web search. The 'availability' column in the table figures out whether the attribute/value is 'hidden' in the dynamic exploratory process, as the 'interaction' means the attribute/value is 'hidden'. Here, about 70% of the information about a recipe is 'hidden' in the interaction process, and is not available on the final page that shows the selected recipe.

Property	Value	Availability
Recipe Name	Spanish Chicken and Rice	
Courses	Stews	Interaction
Courses	Main Dish	Interaction
Main Ingredients	Poultry	Interaction
	Vegetables	Interaction
	Chicken	Interaction
	Meat	Interaction
	Rice	Interaction
	tomatoes	Interaction
	short-grain-rice	Interaction
	pasta-rice-and-grains	Interaction
Preparation	Inexpensive	Interaction
	Served Hot	Interaction
	less 4 Hours	Interaction
	Stove Top	Interaction
	Easy	Interaction
Cuisines	Spanish	Interaction
	European	Interaction
Occasion	Comfort Food	Interaction
	Savory	Interaction
	Potluck	Interaction
	Spread	Interaction
	Fall	Interaction
Dietary	Kid-Friendly	Interaction
	low-sodium	Interaction
	low-calorie	Interaction
	low-carb	Interaction
Calories	768.5	
Calories from Fat	344	
Sodium	176.1 mg	
Cholesterol	167.6 mg	
Potassium	899.0 mg	
Magnesium	82.5 mg	
Carbohydrate	60.8 g	
Protein	43.3 g	

Table 4: incomplete data

C. ALGORITHMS FOR THE ONLINE APPROACH

Algorithm 1 outlines the `OnlineClustering` procedure. For a given list page x, the algorithm finds the query that x belongs to, and outputs a triple $\langle O, Q, C\rangle$, which are the entities (O) that satisfy the query, the query links (Q) on the list pages that belong to the query, and the set of list pages (C) that belong to the query. It is a recursive depth-first-search procedure, and it stops when no list page on the next level is considered similar to the current page.

Algorithm 1: OnlineClustering

Input: A list page x we have not seen before
Output: Set of entity links O, Set of query links Q, Set of list pages C

```
C ← {x} ;
P_x ← all hyper links on x ;
l_x ← links in P_x that point to list pages;
foreach t ∈ l_x do
    y ← page pointed to by t;
    P_y ← all hyper links on y ;
    l_y ← links in P_y that point to list pages;
    if y ∉ C ∧ |l_x ∩ l_y| / |l_x ∪ l_y| > 1 − ε then
        ⟨O_t, Q_t, C_t⟩ ← OnlineClustering(y) ;
        C ← C ∪ C_t ;
    end
end
O ← Entity links on pages in C ;
Q ← Query links on pages in C ;
return ⟨O, Q, C⟩ ;
```

Here, the similarity is measured by the Jaccard coefficient, and we use a small ϵ to tolerate certain error.

Algorithm 2 describes the `OnlineQueryDiscovery` proce-

Algorithm 2: OnlineQueryDiscovery

Input: A list page r we have not seen before; the query forest F we have discovered so far
Output: query forest F

$S \leftarrow \emptyset$; /* S is a stack that stores crawling tasks*/;
Push $\langle r, \{\}\rangle$ to S ;
while *S is not empty* **do**
    Pop $\langle p, A\rangle$ from S ;
    /* Discover the current query: */;
    $\langle O, L, P\rangle \leftarrow CrawlAndCluster(p)$;
    /* Initialize buffer to record drill-down queries: */;
    $Q \leftarrow \emptyset$; ;
    /* Discover drill-down queries of the current query */;
    foreach $l \in L$ **do**
        $a \leftarrow$ attribute and value associated by l ;
        $p' \leftarrow$ list page l points to ;
        $\langle O', L', P'\rangle \leftarrow CrawlAndCluster(p')$;
        $A' \leftarrow A \cup \{a\}$;
        if *$O' \subseteq O$ and $A' \notin Q$* **then**
            Add $O \xrightarrow{a} O'$ to F_{tmp} ;
            Add A' to Q ;
            Push $\langle p', A'\rangle$ to S ;
        end
    end
    if *exisits $u \in F$ such that $u.O = O$ and $u.Q = Q$* **then**
        /* the query is equvilent to an existing query in F */;
        return ;
    end
    $F \leftarrow F \cup F_{tmp}$;
end

dure. Unlike Algorithm 1, it is a breadth-first procedure. It first discovers the current query by invoking `OnlineClustering`, then it uses the query links to obtain its drill-down queries (i.e., queries that are more specific by having one more attribute/value pair). At the end of each level, we check if the DAG forest F already contains a query that is equivalent to the current query. Two queries are considered equivalent if i) ntheir results are the same, and ii) the set of drill-down queries are the same.

D. MORE SURVEY RESULTS

Table 5 and 6 present the detailed results of the survey on 127 exploratory web sites. For each site, we investigate three properties: i)whether the entity pages of the site contain all of the attributes of the entities, ii) can our approach handle the site, and iii) does the site need keyword input to generate list pages. These three properties are the column 'Entity pages contain complete information', column 'can be handled' and 'need keyword input' respectively.

Web Site	Type (Domain)	Entity pages conatin complete information	Can be handled	Need keyword input
www.informatik.uni-trier.de/ ley/db	Digital Library	No	Yes	No
www.computer.org/portal/web/csdl	Digital Library	No	Yes	No
portal.acm.org	Digital Library	No	Yes	No
answers.yahoo.com	Education	Yes	Yes	Yes
wiki.answers.com	Education	No	Yes	Yes
www.msdn.com	Education	Yes	Yes	No
www.answerbag.com	Education	Yes	Yes	Yes
zhidao.baidu.com	Education	Yes	Yes	Yes
wenda.tianya.cn	Education	No	Yes	Yes
books.google.com	Education	No	Yes	Yes
www.ehow.com	Education	Yes	Yes	Yes
www.youtube.com	Entertainment	No	Yes	No
www.youku.com	Entertainment	No	Yes	No
movies.xunlei.com	Entertainment	No	Yes	Yes
www.pogo.com	Entertainment	No	Yes	Yes
video.google.com	Entertainment	No	Yes	No
www.flickr.com	Entertainment	Yes	No	Yes
www.foodnetwork.com	Food	No	Yes	No
allrecipes.com	Food	No	Yes	No
www.kraftrecipes.com	Food	No	Yes	No
www.food.com	Food	No	Yes	Yes
food.yahoo.com	Food	Yes	Yes	Yes
www.epicurious.com	Food	No	Yes	No
www.cdkitchen.com	Food	No	Yes	No
www.bettycrocker.com	Food	No	Yes	No
www.drugstore.com	Health	No	Yes	Yes
yahoo.match.com	Lifestyle	No	Yes	Yes
www.ganji.com	Real Estate	No	Yes	Yes
www.soufun.com	Real Estate	No	Yes	Yes
www.haozu.com	Real Estate	No	Yes	Yes
www.forrent.com	Real Estate	No	No	No
www.frontdoor.com	Real Estate	No	Yes	Yes
realestate.yahoo.com	Real Estate	No	Yes	No
www.realtor.com	Real Estate	No	Yes	No
Trulia.com	Real Estate	No	Yes	No
www.homes.com	Real Estate	No	Yes	No
www.ziprealty.com	Real Estate	No	Yes	No
www.apartmentguide.com	Real Estate	No	Yes	No
www.menupages.com	Restaurant Search	Yes	Yes	Yes
www.restaurant.com	Restaurant Search	No	Yes	Yes
www.restaurantrow.com	Restaurant Search	Yes	No	No
www.zagat.com	Restaurant Search	No	Yes	
dine.com	Restaurant Search	Yes	No	Yes
www.dineout.co.nz	Restaurant Search	Yes	No	Yes
www.eatability.com.au	Restaurant Search	Yes	No	Yes
www.dianping.com	Restaurant Search	Yes	Yes	Yes
www.koubei.com	Restaurant Search	No	Yes	Yes
www.bluefly.com	Shopping	Yes	Yes	Yes
www.urbanoutfitters.com	Shopping	Yes	Yes	Yes
www.torrid.com	Shopping	Yes	Yes	Yes
www.charlotterusse.com	Shopping	Yes	No	Yes
piperlime.gap.com	Shopping	Yes	No	Yes
www.finishline.com	Shopping	Yes	No	Yes
www.neimanmarcus.com	Shopping	Yes	Yes	Yes
www.talbots.com	Shopping	Yes	Yes	Yes
www.paulfredrick.com	Shopping	Yes	Yes	Yes
www.undergear.com	Shopping	Yes	No	Yes
www.ebags.com	Shopping	Yes	Yes	Yes
www.anacondasports.com	Shopping	Yes	Yes	Yes
www.altrec.com	Shopping	Yes	Yes	Yes
www.landsend.com	Shopping	Yes	No	Yes
www.skechers.com	Shopping	No	Yes	Yes
www.allheart.com	Shopping	No	Yes	Yes
shop.pacsun.com	Shopping	No	Yes	Yes
oldnavy.gap.com	Shopping	No	No	Yes
www.llbean.com	Shopping	No	Yes	Yes
shop.nordstrom.com	Shopping	No	Yes	Yes
www.payless.com	Shopping	No	Yes	Yes
www.rustyzipper.com	Shopping	No	No	Yes
www.alight.com	Shopping	No	Yes	Yes
www.lavintage.com	Shopping	No	Yes	Yes
search.80stees.com	Shopping	No	Yes	Yes

Table 5: Exploratory sites

Web Site	Type (Domain)	Entity pages conatin complete information	Can be handled	Need keyword input
www.brooksbrothers.com	Shopping	No	No	Yes
www.casualmale.com	Shopping	No	No	Yes
www.bootbarn.com	Shopping	No	No	Yes
www.shoplocal.com	Shopping	No	Yes	Yes
www.360buy.com	Shopping	No	Yes	Yes
www.dangdang.com	Shopping	No	Yes	Yes
www.shopping.com	Shopping	No	Yes	Yes
www.ebay.com	Shopping	No	Yes	Yes
www.t-mobile.com	Shopping	No	Yes	Yes
www.ecost.com	Shopping	No	Yes	Yes
www.tigerdirect.com	Shopping	No	Yes	Yes
www.jr.com	Shopping	No	Yes	Yes
www.compuplus.com	Shopping	No	No	Yes
shopping.yahoo.com	Shopping	No	Yes	Yes
www.newlook.com	Shopping	No	Yes	Yes
www.overstock.com	Shopping	No	Yes	Yes
www.become.com	Shopping	No	Yes	Yes
www.toysrus.com	Shopping	No	Yes	Yes
www.kelleyfurniture.com	Shopping	No	Yes	Yes
www.zappos.com	Shopping	No	Yes	Yes
www.coggles.com	Shopping	No	Yes	Yes
www.ae.com	Shopping	No	Yes	Yes
www.abercrombiekids.com	Shopping	No	Yes	Yes
www.gap.com	Shopping	No	Yes	Yes
www.amazon.com	Shopping	No	Yes	Yes
www.footsmart.com	Shopping	No	Yes	Yes
www.famousfootwear.com	Shopping	No	Yes	Yes
www.softmoc.com	Shopping	No	Yes	Yes
www.anntaylor.com	Shopping	No	Yes	Yes
www.rustyzipper.com	Shopping	No	Yes	Yes
www.ties2pillows.com	Shopping	No	Yes	Yes
www.thomaspink.com	Shopping	No	Yes	Yes
www.shoebuy.com	Shopping	No	Yes	Yes
www.handbagsworld.com	Shopping	No	Yes	Yes
www.designer-handbags-city.com	Shopping	No	Yes	Yes
www.taobao.com	Shopping	No	Yes	Yes
www.buy.com	Shopping	No	Yes	Yes
www.shoebacca.com	Shopping	No	Yes	Yes
www.hotels.com	Travel	No	Yes	No
travelb.priceline.com	Travel	Yes	No	No
book.bestwestern.com	Travel	Yes	No	No
hotel.qunar.com	Travel	No	Yes	Yes
hotels.ctrip.com	Travel	No	Yes	Yes
www.elong.com	Travel	No	Yes	Yes
www.aa.com	Travel	No	Yes	No
travel.yahoo.com	Travel	No	Yes	Yes
www.travelocity.com	Travel	No	Yes	No
www.orbitz.com	Travel	No	Yes	No
www.cheaptickets.com	Travel	No	Yes	No
www.tripadvisor.com	Travel	No	Yes	No
www.hotwire.com	Travel	No	Yes	No
www.kayak.com	Travel	No	Yes	No
www.yellowpages.ca	Yellow Page	No	Yes	No
www.yellowpages.com	Yellow Page	No	Yes	No
www.yellowpages-china.com	Yellow Page	No	Yes	No

Table 6: Exploratory sites