Link-based Hidden Attribute Discovery for Objects on Web

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

[1]National University of Defense Technology, Changsha, China
[2]Microsoft Research Asia, Beijing, China
[3]Microsoft Research, Mountain View, California, USA
{huangjiuming, yanjia}@nudt.edu.cn
{haixunw,arielf}@microsoft.com

ABSTRACT

Information extraction from the Web is of growing importance. Objects on the Web are often associated with many attributes that describe the objects. It is essential to extract these attributes and map them to their corresponding objects. However, much attribute information about an object is hidden in the dynamic user interaction and is not on the Web page that describes the object. Existing information extraction approaches focus on getting information from the object Web page only, which means a lot of attribute information is lost. In this paper, we study the dynamic user interaction on exploratory search Websites and propose a novel link-based approach to discover attributes and map them to objects. We build an exploratory search model for exploratory Web sites, and we propose algorithms for identifying, clustering, and relationship mining of related Web pages based on the model. Using the unsupervised method in our approach, we are able to discover hidden attributes not explicitly shown on object Web pages. We test our approach on two online shopping Websites. We achieve high precision and recall: For entirely crawled Web sites the precision and recall are 98% and 97% respectively. For randomly crawled (sampled) Web sites the precision and recall are 98% and 80% respectively.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *data mining*; H.3.3 [**Information Systems**]: Information Search and Retrieval – *retrieval models*

General Terms

Algorithms, Experimentation, Theory

Keywords

Attribute discovery, attribute labeling, exploratory search, information extraction, query link.

EDBT 2011, March 22-24, 2011, Uppsala, Sweden.

[*] The work was done at Microsoft Research Asia.

1. INTRODUCTION

The Web contains rich information about a large variety of objects such as people, products, locations, etc. Many applications rely on information extraction techniques that convert information on the Web into structured, machine readable format. Current techniques focus on extracting information from content on given Web pages. For instance, given a Web page about a product on an e-commerce Website, much work [1-5] focused on obtaining such information as the name, brand, price, and other properties, of the product. However, there are many situations where a lot of useful information about the product do not appear on the final page that shows the product, but rather, in the dynamic user interaction process that leads the user to the page.

We illustrate such situations using an example. Figure 1 is a product page from Zappos.com, a well known online shopping Website. The page shows an article of clothing known as "Tunnel Vision Shirt". The information on the page includes the name, brand, price, color, size, width, and SKU of the product.

Figure 1. Product page.

However, much important information about the "Tunnel Vision Shirt" is not on the Web page in Figure 1. In Table 1, we show a total of 15 properties of the product, all of which is obtained from the Zappos.com Website, but only less than half of the information (marked "static") is found on the page in Figure 1.

The rest of the information, which we label as "interactive" or "hidden", do not exist on the Web page. Instead, it is embodied in the interactive process with which the user explores the Website and finds the product. Thus, the problem is, how can we find such information about the product?

Table 1. A complete property list of "Tunnel Vision Shirt"

Property	Value	Availability
Product Type	Clothing	Interaction
Gender	Girls	Interaction
Category	Tops	Interaction
Subcategory	Long sleeve tops	Interaction
Top Type	Blouse	Interaction
Price Boundary	$50.00 and Under	Interaction
Theme	Casual	Interaction
Collar Type	Spread	Interaction
Width	Apparel	Interaction & Static
Color	Gloxinia	Interaction & Static
Brand	Roxy Kids	Interaction & Static
Size	XL	Interaction & Static
Name	Tunnel Vision Shirt	Static
Price	$32.00	Static
SKU	7690411	Static

One example of such interaction is the so-called *faceted search* [6]. Figure 2 shows the example of faceted search on Zappos.com. For example, the "Tunnel Vision Shirt" can be found among the products shown as a result after the user clicks on "Tops" in "Category", however, the final page of the product in Figure 1 does not contain the information "Category = Tops." From our studies, we found that over 80% of product information is embodied in user interaction, and less than 50% exists on the object pages or the final product pages (some interactive information is also available on final product pages).

Zappos.com and e-commerce are not alone in having this problem. More and more Websites support the emerging Exploratory Search model [7], which combines querying and browsing in Web surfing. Compared with typical keyword search, exploratory search supplies additional hierarchical or multi-dimensional browsing options for users who do not know much about their goals. There are a broad range of applications of exploratory search. For example, almost every online shopping Website uses faceted search to give users a variety of categorical search options; digital libraries, such as ACM Digital Library and DBLP, supply users with multiple dimensions in search including publication years, authors, venues, etc.; collaborative tagging [8] systems such as Delicious.com use tags as useful cues for people to discover topics that are relevant. Thus, much information about an object of interest is being used in exploratory search for goal query in search and exploration, and is not presented explicitly on

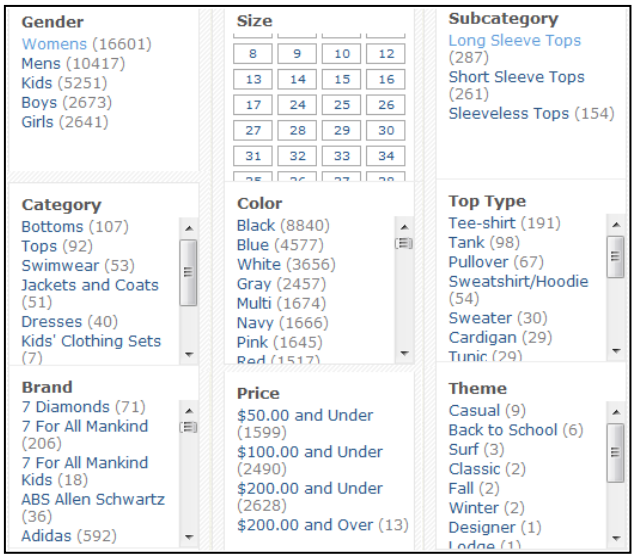

Figure 2. Faceted search on zappos.com

the target Web pages. As a result, traditional information extraction techniques that ignore the dynamic user exploration process will not be able to obtain complete information about the objects of interest.

The Naïve Approach. In order to find the complete set of attributes and their values, we can take the following naïve approach:

- First, collect all the interaction hyperlinks. A good place to start is the faceted search box such as the one shown in Figure 2.
- Second, for each hyperlink, find all the objects that the hyperlinks point to.
- Third, for each hyperlink, find what attribute/value it represents.

Unfortunately, each step requires a lot of user supervision. For instance, in the first step, it is difficult to automatically identify hyperlinks that correspond to exploratory search/browsing. For a given Website, say zappos.com, we may discover that interactive links satisfy a certain pattern containing the string "www.zappos.com/search". Thus, it requires us to perform such kind of investigation on Website.

The second step presents even more problems. Each interactive hyperlink corresponds to a search criterion. However, the objects that satisfy a search criterion may not be in the single page the hyperlink points to, and not every object on the single page satisfies the search criterion (the page may contain objects related to the search, or objects that are in promotion or under highlight). Thus, in order to find the complete set of objects that satisfy the search criterion, we need to tell the system what are the patterns that group multiple pages satisfying a search criterion together, and what region of the page contains objects that satisfy the search query, etc. Thus, we need a lot of user supervision so that we can provide a set of templates to the system. Even if the Websites do not change their templates frequently, it is a tedious

task to find and maintain such templates for a large number of Websites.

Our Contributions. We present an unsupervised approach for automatically discovering *interactive* or *hidden* attributes for objects in a variety kinds of Websites that support exploratory search/browsing. Our approach achieves an average precision of 98% and an average recall of 97% for Web sites that support exploratory search/browsing.

More specifically,

- We introduce a model for exploratory search/browsing. Without exception, every Website uses certain templates to present results that satisfy a search criterion. Although the templates are different for different Websites, they implement a similar set of semantics. For example, the concepts of "roll-up queries" and "drill-down queries" are universal. Our model focuses on uncovering such semantics.
- We introduce an unsupervised method for crawling and information extraction. In particular, we propose algorithms to automatically group web pages that correspond to a single search criterion, and we propose algorithms for automatic query identification, including roll-up and drill-down link detection. Since the algorithms are based on the internal semantics of exploratory search/browsing, there is no need to provide handcrafted templates for each individual Website, which makes our approach general.

Paper Organization. The rest of the paper is organized as follows. Section 2 models the exploratory search Websites and defines attribute discovery task based on the model. Section 3 discusses a naïve approach. Section 4 presents our approach based on the proposed model. Section 5 presents the experiment, including the construction of data set and the evaluation for the experimental results. Section 6 brings this paper to a conclusion and discusses future works.

2. EXPLORATORY SEARCH/BROWSING

Exploratory search/browsing combines keyword search and multidimensional browsing to help users narrow down on the objects they are looking for. Unlike traditional navigation, in exploratory search/browsing, a page often corresponds to a search of a fixed form, and many hyperlinks on the page correspond to new searches that are related to the current search.

Websites that provide exploratory search/browsing have many common characteristics. Our goal is to discover the semantics underneath these common characteristics so that we can use the semantics to guide web crawling and information extraction.

2.1 Queries and their Relationships

We first formalize the concept of faceted search. We define a faceted search as a set of <*attribute, value*> pairs. For example, a search represented by {<*type, 'Clothing'*>, <*gender, 'Girls'*>} finds girls' clothing.

We can easily express such queries in SQL, for example:

q : **SELECT** *
FROM products
WHERE Type = 'Clothing' **AND** Gender = 'Girls';

If we know the <*attribute, value*> pairs that correspond to a query *q*, and we know the objects that satisfy *q*, then we know these objects must have the values for the attributes specified by *q*. Thus, to find the complete set of attribute values for all objects, we need to find:

1. the complete set of queries (including the <*attribute, value*> pairs that correspond to each query);
2. objects that satisfy each query (we address this problem in Sec 3).

The first problem is difficult. The Website does not list all the queries, nor does it specify the set of <*attribute, value*> pairs for each query.

In our work, we discover queries through their relationships. In faceted search, for example, queries form "drill-down" and "roll-up" relationships among themselves. As an instance, consider the query *p* below, whose **WHERE** condition is a superset of that of *q*:

p : **SELECT** *
FROM products
WHERE Type = 'Clothing' **AND** Gender = 'Girls'
AND Category = 'Tops';

We say *p* is a *drill-down query* of *q* and *q* is a *roll-up query* of *p*. Instead of knowing the <*attribute, value*> pairs for a query *q*, all that we know is the *hypertext* of the drill-down link between *q* and *p*, that is, label(*q*→*p*) = <Category, Tops>, that is, query *p* narrows down on query *p* by specifying an additional <*attribute, value*> pair.

In the rest of the paper, we focused on solving the above two problems.

2.2 Page Classification

Before we address the problem of finding queries and objects that satisfy the queries, we first build a model for exploratory Web search. Our first job is to classify pages on an exploratory search site into 3 types: ***Result Pages***, ***Object Pages***, and ***Unrelated Pages*** (which are pages that are neither result pages nor object pages).

A faceted search returns a **Result Page** from which a user can explore objects that satisfy the search criterion. Note that a search usually corresponds to a set of result pages instead of a single page. Consider the web page in Figure 3. It is the result of faceted search *q*: "Clothing -> Girls" on zappos.com, that is, it is the page shown after the user clicks on "Clothing" and then "Girls" in faceted search. The result page contains many hyperlinks, including links to a partial set of objects (4 objects) that satisfy the query, hyperlinks (called *pagination links*) to other result pages for query *q*, and hyperlinks that correspond to other faceted searches (related or unrelated to the current search). Below, we analyze hyperlinks on the result page, and we classify them into six categories:

- **Drill-down query links**: which represent searches that narrow down user's selection. For example, if the user clicks on 'Tops', he will trigger a new faceted search *p*, which is a drill-down query of the current query *q*.

Figure 3. Search result page.

- **Roll-up query links**: which represent searches that generalize the current search. For example, if the user clicks on 'Girls', he will trigger a new faceted search whose condition contains 'Type=Clothing' only, which is a roll-up query of the current query *q*. Note that in many Websites, drill-down queries and roll-up queries may form a complicated hierarchy and appear in the same area on the result page.
- **Unrelated query links**: which represent searches that are unrelated to the current search. For example, links in the top navigation bar, such as 'SHOES', 'CLOTHING', …, and links in the alphabetical brand search, as well as possible popular search options, are unrelated query links.
- **Object page links**: which point to objects that satisfy the current search. In our example, we have 4 object links;
- **Pagination links**: which point to other result pages for the current search. Usually, it does not have links to all the pages, but rather, it provides an iterative mechanism for users to page through all the result pages.
- **Other links**: which are any hyperlinks that do not fall into the above 5 categories. These include links such as those that lead to "My Cart", account management, objects in promotion, etc.

The drill-down query links, roll-up query links, and unrelated query links represent other faceted searches, while object page links and pagination links lead to objects that satisfy the current search. Note that, given a page on an arbitrary Website, it is most likely impossible for the machine to automatically identify and differentiate among the 5 types of links. Instead, we need to take a supervised approach, that is, we need to tell the system which part of the page contains which types of links, and the syntactical patterns these links exhibit. This means we need to build customized crawling for every Website, which is certainly not scalable.

Besides result page, another important type of page is **Object Page**, which contains information about a single object (See Figure 1 for an example). *Objects* are basic entities, such as products on shopping Websites, books on digital library sites, bookmarks on social bookmarks Websites. Each object is described by a set of attributes, such as price, size and color for a product, title, author and publication for a book, category, tag, source for a bookmark. These form facets for search. In this paper, for simplicity of discussion, we assume each object is represented by one and only one object page.

We denote pages that are neither result pages nor object pages as **Unrelated Pages**. These include pages for account management, online transaction, site description and advertisement, etc. Note that it is not trivial for machines to automatically differentiate the 3 page types without resorting to a supervised approach.

2.3 Semantics Embedded in Link Structures

Our goal is derive a complete set of attribute values for each object. Since attribute values are embedded in queries, we need to uncover the mapping between queries and sets of objects that satisfy the query. A naïve approach needs to find, for each Website, the syntactic patterns that encode the mapping between queries and objects. This requires a lot of manual work.

We focus on uncovering the relationships between queries and objects by mining the hyperlinks among i) faceted searches, ii) result pages, and iii) object pages. We accomplish this without relying on syntactic patterns, and thus, there is no manual work involved. Specifically, we represent an exploratory search/browsing Website by a graph $<P, L>$, where P is the vertex set (each vertex represents a web page), and L is the edge set (each edge represents a hyperlink from one web page to another

web page). Besides the visible links between Web pages, there are also invisible links between queries and result pages.

Figure 4 illustrates the relationship among queries, result pages, and object pages in our model. All the attributes constitute an attribute space, which is effectively a query space, as a search consists of a set of attribute and value pairs. Here we see that q1 is contained by q2, which means q1 has fewer attributes than q2,

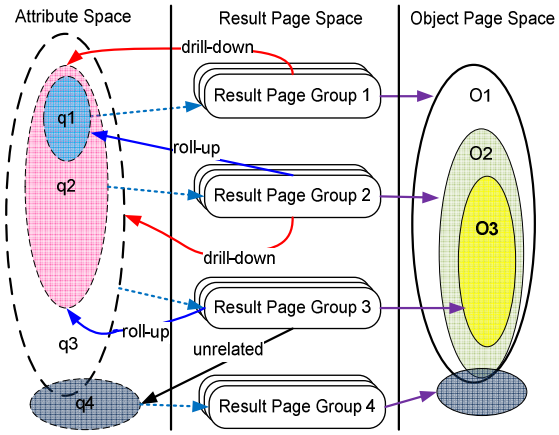

Figure 4. Relationship among queries, result pages, and object pages.

and is thus a roll-up (more general) query of q2. Furthermore, each query maps to a set of result pages, which in turn contain all the objects that satisfy the query. Note that if a query corresponds to more than one result page, these result pages will be connected using pagination links. But on any result page, the pagination links may not be complete, that is, a result page may only contain links to a previous or a next result pages, instead of the whole set of result pages of the query. Thus, recovering the whole set of result pages may require some iterative steps if we can identify the pagination links, which itself requires some supervised learning. In our approach, we cluster pages based on their internal hyperlink structures and avoid any supervised learning cost.

Formally, we define the links among queries, result pages, and object pages as functions shown below:

Definition 2.1 (Result function): $Q \xrightarrow{\varphi} 2^R$ *maps a query to a group of result pages.*

Definition 2.2 (Object Function): $Q \xrightarrow{\omega} 2^O$ *maps a query to a set of object pages.*

For any query q, it is clear that $\omega(q)$, the objects that satisfy q, are listed on result pages $\varphi(q)$.

There are some interesting properties in the link structures on an exploratory search/browsing Website.

Property 2.3: *A result page belongs to one and only one query.*

That is, even if two queries q1 and q2 have exactly same results (a same set of objects satisfies q1 and q2), they have a separate set of result pages. The result pages are unique because besides the hyperlinks to the objects, they also contain links related to the query, including, for example the roll-up and drill down queries of q1 and q2. Since q1 and q2 are different, their roll-up and drill down queries are different, which means their result pages are different. Thus, we can define an inverse function that maps result pages back to queries:

Definition 2.4 (Inverse result function): *Since each result page belongs to one and only one query, so function* $2^R \xrightarrow{\varphi^{-1}} Q$ *exists.*

We use this property to group result pages by queries (Section 4).

The second important property is concerned with the query links on result pages. On a result page, there are 6 types of links, namely, roll-up query links, drill-down query links, unrelated query links, object page links, pagination links, and other links (Section 2.1). For example, Figure 4 shows that pages in Result Page Group 1, which correspond to query q1, contain drill-down links to q2 (as defined, $q1 \subset q2$), while pages in Result Page Group 2, which correspond to query q2, contain roll-up links to q1. Result Page Group 2 and Result Page Group 3 have similar interlinks, and so on. Result Page Group 4 does not have roll-up or drill down relationships with q1, q2, or q3, but still, it may contain links to q1, q2, or q3 as unrelated query links, vice versa for pages in Result Page Group 1, 2, and 3.

Consider two pages *p1* and *p2* in the same result page group. Clearly, the two pages will have different pagination links and object page links. However, the roll-up and drill-down links on *p1* and *p2* are the same, that is, we have the following property:

Property 2.5 *Let q be a query, for any two pages in* $\varphi(q)$, *that is, for any two pages in the same result page group of query q, the roll-up and drill-down links on the two pages are most likely to be the same.*

The property holds because roll-up queries and drill drown queries depend on the current query only. Since pages in the same result page group are generated by the same query, they will have the same set of roll-up and drill-down links. This is of course based on the assumption that each result page presents all of possible drill-down links and roll-up links for the current query q. This assumption is true for all the exploratory search/browsing Websites we have surveyed.

Besides the above two properties, there are some other properties which are important to the correctness of our approach. These include:

- Queries are deterministic, that is, the set of result pages of a query q is fixed (unless objects are added into or removed from the database). In other words, though the execution of query q may be triggered by different hyperlinks or at different time, q always generates the same result page group;
- Query results are complete, that is, a query retrieves all the objects that satisfy the query. In other words, if an object has all the attribute values specified by the query, this object is certain to be retrieved by this query;
- All result pages are accessible, that is, starting from any result page, users can follow pagination links to access all the result pages in the same result page group.

3. UNSUPERVISED APPROACH

In this section, we present our approach of finding hidden attribute values for objects on exploratory search Websites. The approach is general as it does not rely on Website specific templates. We first give an overview of our approach, and then describe three algorithms, namely *identifying page types*, *clustering result pages*, and *detecting relationship between queries*.

3.1 Overview

Our goal is to find attribute values for each object. The attribute values are hidden or embedded in queries. So our approach is to find: i) the complete set of queries; ii) the <*attribute, value*> pairs that correspond to each query; and iii) the objects that satisfy each query. Let o be an object, and let $Q(o)$ be the set of queries that o satisfies. Then, o's attribute values are the unions of the set of <*attribute, value*> pairs of all the queries in $Q(o)$.

However, queries are not first class citizens on exploratory search Web sites. In our approach, we represent queries using their result page groups. Let q be a query, we use $\varphi(q)$, the set of result pages correspond to q, to represent q. Thus, the result page groups act as a bridge between the queries and the objects.

Specifically, we adopt the following steps to carry out the above task:

(1) **Crawling and preprocessing.** We first crawl the entire Website. We identify each page using its URL and extract all hyperlinks from the pages.

(2) **Identifying the type of each page.** We classify Web pages into three types: result, object and unknown.

(3) **Clustering result pages for queries.** This step clusters result pages into groups, each of which is for a query. The clustering method takes advantage of the observation that result pages for a same query co-cite a largely same set of result pages.

(4) **Detecting relationship between queries.** As we have discussed, hidden attributes exist on the query links (hypertext of the link from a roll-up query to a drill-down query). For each query link, say X, we create an attribute, using the hypertext of X as attribute value and the information of upper level HTML element of X as attribute name. There are a lot of mature technologies to find the upper level HTML element of a hyperlink, such as [25-27].

(5) **Assigning attributes to objects.** The hidden attributes of an object are recovered as the union of the <*attribute, value*> pairs of all the queries that the object satisfies.

The challenges in above steps are identifying pages' types, clustering result pages and detecting relationship between queries. If these three steps can work generally and unsupervised, the total process will run without any template. The rest of this section will describe our algorithms for these key steps.

3.2 Page Type Identification

For an input Website, we want to find the relationships among its result pages and object pages. To do this, first we need to classify all of its pages into three types: result pages, object pages, and unknown pages.

It is usually very easy to use a specific pattern to identify an object page. For example, on an e-commerce Website such as Zappos.com, each object page contains a label "Add to shopping cart." Certainly, this pattern is site specific. However, since it is very easy to identify such a pattern, and the pattern is extremely effective, we use hand-coded patterns to identify object pages. This is the only site-specific pattern we use in our work. Alternatively, we can use Web page classification methods based on machine learning techniques, such as [12] and [13], for this task as HTML structures of object pages on the same Website are quite similar.

Our concern is then how to differentiate result pages from unknown pages. We take a greedy approach, and classify any non-object page as a result page as long as it contains one hyperlink to an object page. Although theoretically this may lead to some issues in clustering result pages, we find the impact is negligible. In our experiments, we analyze the effect in detail.

Algorithm 1 shows our page type identification algorithm. It is a two phase algorithm. In the first phase, it identifies the pages contain "Add to shopping cart" label as object type. In the second phase, it identifies the remnant pages as result type or unknown type.

Algorithm 1 *Page Type Identification*

Input: A Website $W=<P, L>$
Output: *a map from pages to types*

```
1:  for  p ∈ P do
2:       if p contains "Add to shopping cart" then
3:            Identify p as object page;
4:       else
5:            Identify p as unknown page;
6:       end if
7:  end for
8:  for each unknown page p do
9:       if p contains a link to x and x is object page then
10:           Identify p as result page;
11:      end if
12: end for
```

Obviously, Algorithm 1 only need scan every Web page twice, so the time complexity of it is $O(|P|)$.

3.3 Link-based Result Page Clustering

We explore hyperlink structures to cluster result pages into result page groups. We introduce a distance measure based on pages' co-citation information, and then we use a single-pass clustering algorithm based on the distance measure.

According to Property 2.5, if two result pages belong to the same query, their drill-down links and roll-up links are the same. Besides drill-down links and roll-up links, unrelated query links (links to objects under promotion, for example) on result pages that belong to the same query are often the same as well (unrelated queries are either site-specific or query-specific). In other words, if two result pages belong to the same query, they *co-cite* a lot of result pages.

Inspired by the above observation, we introduce a *distance* measure to describe the co-citation situation of two pages.

Definition 3.1 (Distance of links): *Let R be the set of all result page. Let* $\tau(p) = \{p' \in R \mid p \rightarrow p' \in L, p' \in R\}$ *denote the set of result pages p points to. For* $p \in R$ *and* $p' \in R$ *, let* $\delta(p, p')$ *denote their distance, which is defined as the symmetric difference between* $\tau(p)$ *and* $\tau(p')$ *, that is*

$$\delta(p, p') = |(\tau(p) - \tau(p')) \cup (\tau(p') - \tau(p))| \quad (1)$$

In other words, the larger the distance between two pages, the less likely they belong to the result of the same query. Intuitively, if *p* and *p′* belong to the same query *q,* then the query links of *p* and the query links of *p′* should be largely the same. Their difference is mainly the links to the objects that belong to *p* and *p'* respectively, and the pagination links. Thus, *p* and *p′* should have many common links, so $\delta(p, p')$ is small. On the other hand, if *p* and *p′* belong to two different queries, say, page *p* belongs to query $\theta(p)$ and *p′* belongs to query $\theta(p')$, $\theta(p) \neq \theta(p')$, then the distance between *p* and *p'* will tend to be large. This is because the drill-down and roll-up query links depend on the query, so the result pages pointed to by the drill-down and roll-up query links of *p* are quite different from those of *p′*. Furthermore, the pagination links of *p*, which point to other result pages for $\theta(p)$, are totally different from the pagination links of *p′*.

Based on the nature of the above distance function, we introduce a threshold to indicate whether two pages belong to the same query or not. Formally, let ***d*** be the threshold, let *p* and *p′* be two result pages, the assumption is:

$$\forall_{p,p' \in R} \left(\delta(p, p') < d \Leftrightarrow \theta(p) = \theta(p')\right) \quad (2)$$

Algorithm 2 clusters result pages according to Eq (2). The output is a set of result page groups (***RPG***). In the main loop, we check every result page. If a result page, say *p*, has not been added to any group in *RPG*, we create a new group using *p* as a seed page, then we group neighboring pages starting from *p*, and add the new group to *RPG* at the end. The *GroupNeighboringPage* function walks from a seed page to all pages which are neighboring with their previous pages. That is, for all pages whose distance from current page is smaller than *d*, say *p′*, the function add *p′ to* the group of current page and recursive call itself using *p′* as seed page.

Algorithm 2 finds the complete set of result pages for each query. To see this, let *q* be a query pointed to by any drill-down link on the web site, and let *Q* be the set of all such queries. The members of *Q* have a one-to-one mapping with members of *RPG*. The reason is as follows. First, Line 2 of the algorithm checks every result page of the input data and line 4 makes sure that every result page will not be added into more than one group, so each result page belongs to one and only one group of *RPG*. Second, line 2 in *GroupNeighboringPage* makes sure only result pages for the same query will be added to a same group. Furthermore, if a query generates more than one result page, each result page will be pointed to by at least one result page for the query using pagination links (easily inferred from the accessibility of result pages). So if two pages are for the same query, they will be clustered into a same group finally. Thus, the queries of *Q* have one-to-one mapping to the groups of *RPG*.

Finally, the time complexity of Algorithm 2 is $O(|P|)$, since every result page is checked once.

Algorithm 2 *Result pages Clustering*

Input: a Website *W*=<*P, L*>
a threshold *d*
Output: the set of result page groups (RPG)
1: *RPG*←∅, *LS*←{*all result pages*};
2: **for** $p \in LS$ **do**
3: **if** $\exists_{s \in RGP}(p \in s)$ **then**
4: **continue**;
5: **end if**
6: *G*←∅, *E*←∅;
7: Add *p* to *g*;
8: GroupNeighboringPage (*G*, *E*, *p*);
9: Add *G* to RPG;
10: **end for**

function GroupNeighboringPage (*G*, *E*, *p*)
1: **for** $p' \in \tau(p) \wedge p' \notin G \wedge p' \notin E$ **do**
2: **if** $\delta(p, p') < d$ **then**
3: Add *p′* to *G*;
4: GroupNeighboringPage (*G*, *E*, *p′*);
5: **else**
6: Add *p′* to *E;*
7: **end if**
8: **end for**

3.4 Detecting Relationship between Queries

Our goal of detecting relationships between queries is to recover <*attribute, value*> pairs for queries. As mentioned, hidden attributes are embedded in query links. A query link essentially associates two queries with specific semantics depending on whether it is a drill-down, a roll-up, or an unrelated query. We discuss how to obtain <*attribute, value*> pairs from different query link types, and then propose an algorithm to select useful query links.

The drill-down and roll-up links encode the relationship between two queries. Let *p* be a drill-down query of *q* (equivalently, *q* is a roll-up query of *p*.) Then, in many cases, the drill-down hyper link from *q* to *p* will contain the <*attribute, value*> information as an anchor text. For instance, assuming a user is browsing a page about women's shoes, then, a drill-down link on that page may have the anchor text "size = 6" or just "6" (we will discuss how to discover the attribute name when only values appear in the anchor text). From the anchor text, we obtain <size, 6> as the attribute value pair that denotes the relationship between the two queries. The case with roll-up link, however, is not always the same. For instance, on Zappos.com, the anchor text on a roll-up link may simply be "remove your selection." As another example, Amazon.com displays a set of query links in a hierarchy, for instance, "Shoes -> Women -> Size -> …" Thus, a single anchor text "Shoes", "Women", or "Size" is not informative, and understanding the hierarchy requires developing handcrafted rules. On the other hand, a query link from an unrelated page (e.g., a page showing a user's account information) may contain the attribute/value pair that defines the query. For instance, the anchor text may simply be "Women's" on a e-commerce site that sells shoes. In summary, exploratory search Websites often

present attributes on the drill-down links and unrelated links to inform uses about the content of the link.

Based on the discussion above, we can see that the anchor texts in drill-down and unrelated links are more reliable. Thus, our goal becomes how to differentiate roll-up links from drill-down and unrelated links.

Based on the query definition, given two queries p and q such that q is the drill down query of p, then the set of objects that satisfy q must be a subset of the objects that satisfy p. That is, the two result sets must have set containment relationship. In reverse, if two object sets, each of which derives from a query, have set containment relationship, the corresponding queries may be connected by a drill-down link or an unrelated link. In other words, if an object set X which derives from a query q_X, is a subset of another object set Y which derives from a query q_Y, then q_X must not be the roll-up query of q_Y. We represent queries using result page groups, and discuss how the subset relation between two object sets affect query links between two result page groups. Formally,

Theorem 3.2: *Let RPG be the set of all result page groups.* $\forall s \in RPG$, $\forall r \in RPG$, $s \neq r$, *if* $\omega(\varphi^{-1}(s)) \subset \omega(\varphi^{-1}(r))$, *then query links between s and r are not roll-up query links.*

Proof: $\forall s \in RPG$, $\forall r \in RPG$, $s \neq r \Rightarrow \varphi^{-1}(s) \neq \varphi^{-1}(r)$ because groups in *RPG* correspond one-to-one with queries. Thus, links among s and r contain no pagination link. All pages in s or r are result pages, so $\forall x \in r$, $\forall y \in s$, x may points to y using (1) drill-down query link, or (2) roll-up query link, or (3) unrelated query link, or (4) no link. If $\omega(\varphi^{-1}(s)) \subset \omega(\varphi^{-1}(r))$, according to the completeness of queries, $\varphi^{-1}(s) \not\subset \varphi^{-1}(r)$ otherwise all object retrieved by $\varphi^{-1}(r)$ will be retrieved by $\varphi^{-1}(s)$. Thus, $\omega(\varphi^{-1}(r)) \subseteq \omega(\varphi^{-1}(s))$. According to the definition of roll-up query, $\varphi^{-1}(s)$ is not a roll-up query of $\varphi^{-1}(r)$. If $x \to y$, the possible cases of x pointing to y remain case (1) and case (3). ■

Another useful and obvious property of relationship between object sets is that if two object sets, each of which derives from a query, are equal, and their corresponding result pages have query links point to each other, the attributes presented by those query links are certain to be common attributes of the two object sets.

In conclusion, if a query link associate query x to query y, and $\omega(y) \subseteq \omega(x)$, the attributes presented by the link will belong to query y. We denote this kind of query link as *useful link*. If we can find out useful links, the mapping from attributes to queries is known.

A naïve approach that detects useful links need to check every query link. For each result page group, say s, we check all query links on pages in s. If a query link points to page in result page group r, we check the subset relationship between $\omega(\varphi^{-1}(s))$ and $\omega(\varphi^{-1}(r))$. Assume the average time cost of checking subset relationship is C. The total cost of naïve approach is as high as $O(|L|*C)$.

Algorithm 3 *Useful link detection*

Input: A Website $W=<P, L>$, *RPG*
Output: Useful link set D

```
1: D ← ∅;
2: for s ∈ GAP do
3:     p ← arbitrary page in s;
4:     for each p → x ∈ L do
5:         r ← result page group of x;
6:         if ω(φ^-1(r)) ⊆ ω(φ^-1(s)) then
7:             Add k to D;
8:         end if
9:     end for
10: end for
```

Algorithm 3 is our pruning method to detect useful links. We prune some repeated check operations in the above simple approach. Property 2.5 and the co-citation phenomenon discussed in Section 3.3 figure out that, if two pages belong to the same query, their query links are the same except pagination links. And the attributes on the query links are certainty, reviewing our goal which is to map attributes extracted from useful links to objects retrieved by the queries, we only need to check arbitrary one page for a result page group.

The time complexity of Algorithm 3 is $O(|RPG|*C)$. Obviously, $|RPG|$ is far smaller than $|L|$.

4. RELATED WORK

A lot of existing work extracts attributes from semi-structure data (e.g., static Web pages). There are three main categories of information extraction technologies: 1) Traditional rule or regular expression based approaches (e.g., [1, 14, 15]). Typically, they are efficient and can achieve high accuracy when the task is controlled and well-behaved, for example, the extraction of price information from Web pages; however, they are not general because they depend on human experts and programmers to design hand-coded patterns for every Website; furthermore, as discussed in [16], they cannot handle noises in a robust manner; 2) Supervised machine learning approaches (e.g., [17, 18]). They are more automatic as they learn extraction rules from manually labeled Web pages; and they are useful in closed domains where human involvement is both essential and available. However, the training is expensive; 3) Unsupervised approaches (e.g., [19, 20]). Unsupervised approaches are more robust to noise and more general for various data sources but usually it is difficult to achieve high accuracy and they often make many assumptions which may not always be satisfied.

Much recent research focuses on information extraction without using templates, and often uses technologies such as ontology and text mining to get better results. For instance, Holzinger et al [21] use ontology for extracting product information from tabular data on Web pages. Its rectangular table model does not make use of the additional structural information that is present in more complex layouts. Other works [2, 4, 22] treat Web information extraction as a classification problem. For example, single-view and multi-view semi-supervised learning algorithms [2, 4] are developed to exploit large amounts of unlabeled data. However, their performance is not quite good since they do not take

advantage of characteristics of Web structure. Recently, extracting information from tables attract a lot of attention. Zhai et al and Gatterbauer et al [24, 25] propose methods that mine data records from pages based on the structure of HTML. They combine tag string match and some visual features to achieve better performance. More recently, Wang et al [3] uses an automatically constructed knowledgebase [30] for large scale information extraction from Web tables.

Lerman et al [26] took advantage of the structure of Websites to automatic extract and segment records from Web tables. The idea of [26] is quite similar to our work. Its approach relies on the common structure of many Websites, which present information as a list or a table, with a link in each entry leading to a detail page containing additional information about that item. But its goal and algorithms are quite different from our work.

Discovering attributes for objects from massive Web data benefits various applications including information retrieval, ontology building and data mining. If we can restructure information of products from Web pages on online shopping Websites, it is easy to build shopping comparison services [27] like shopping.com, or mine high level knowledge for competitive intelligence [28]. If information about papers on DBLP, ACM Digital Library can be understood automatically, faceted search engine for scholar is easy to build. And extracting metadata of social bookmarks from Delicious can facilitate machine understanding the relationship among various concepts, so as to build ontology. Recently WePS [29] campaign which focuses on Web People Search problem has been held three time because the challenges and importance of the problem.

5. EXPERIMENT

This section reports our experiment on the unsupervised approach on two well-known exploratory Website: Zappos.com and Amazon.com. We evaluate the precision and recall on different scale of data and we show that our unsupervised approach is effective with close to 100% recall and precision. We further analyze why our approach cannot achieve close to 100% recall and precision.

5.1 Data Sets

Zappos.com is a typical faceted search Website. It has good design which makes it possible to manually create some templates for the Website, which allows us to extract attributes using the naïve approach (as described in the introduction). We perform the naïve approach on the Zappos data and use the result as a validating dataset.

We crawl the entire Web site of Zappos.com beforehand. The total number of relevant Web pages is 11,955,201 (Web pages that do not have additional information about the products, for example, pages that display products in different order, are filtered). We also derive a "ground truth" dataset by handcrafting templates for Zappos.com and then use the naïve approach to extract products and their attribute values.

Table 2 shows some statistical information about Zappos, including the number of real result pages, objects and total links. We also show the number of pages that contain links to objects but are not related with any queries (we call them *fake result pages* here) and the number of <*object, attribute*> pairs in the data set.

Table 2. The Zappos Dataset

# of result pages	# of objects (products)	# of <object, attribute> pairs	# of fake result pages	# of total links
11831247	124839	2208052	2448	2391035716

Amazon.com is a famous and well designed online shopping Website too. We also crawl Web pages from Amazon.com (mostly pages in the Shoes category) as another data set. The data set has 28,186,473 Web pages after applying the same filtering strategy for Zappos. Table 3 shows the statistical information about the Amazon data set.

Table 3. The Amazon data set

# of result pages	# of objects (products)	# of <object, attribute> pairs	# of fake result pages	# of total links
27515023	634767	1853145	36683	5637294633

5.2 Evaluation Method

We evaluate our experimental results using the conventional precision and recall measure. Recall that our goal is to discover hidden attributes for objects. We define the precision and recall for single object respectively as Eq (3) and Eq (4):

$$recall(i) = \frac{|\, real(i) \cap machine(i) \,|}{|\, real(i) \,|} \tag{3}$$

$$precision(i) = \frac{|\, real(i) \cap machine(i) \,|}{|\, machine(i) \,|} \tag{4}$$

where i is an object, $real(i)$ is the real attribute set of i, $machine(i)$ is the detected attribute set of i.

For recalls and precision on the entire dataset, we sum the single evaluation for each object with a weight, where the weight is the ratio of attributes of the object divided by the total number of real attributes. Equation (5) and equation (6) define the recall and precision for dataset s:

$$recall(s) = \sum_{i \in s} recall(i) \times \frac{|\, real(i) \,|}{\sum_{j \in s} |\, real(j) \,|} \tag{5}$$

$$precision(s) = \sum_{i \in s} precision(i) \times \frac{|\, real(i) \,|}{\sum_{j \in s} |\, real(j) \,|} \tag{6}$$

5.3 Experimental Results

In order to test the performance and the overhead of our unsupervised approach, we test on various subsets of our data sets. In particular, we gauge the effect of the distance threshold in clustering on the precision and recall on small datasets; and we test the precision and recall on big data sets with the detected best parameter. To measure the effectiveness and overhead, we count the number of object-attribute pairs that can be detected from datasets of various sizes.

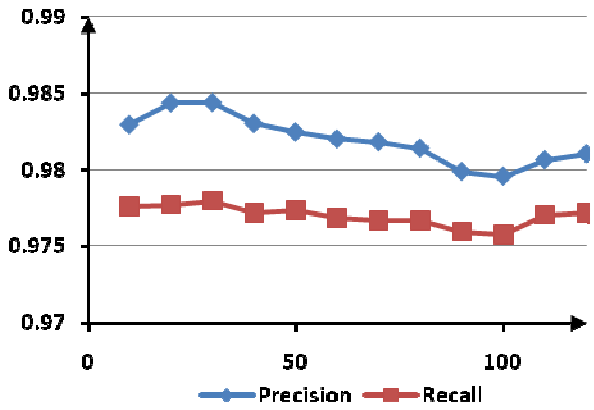

Figure 4. Performance on Zappos

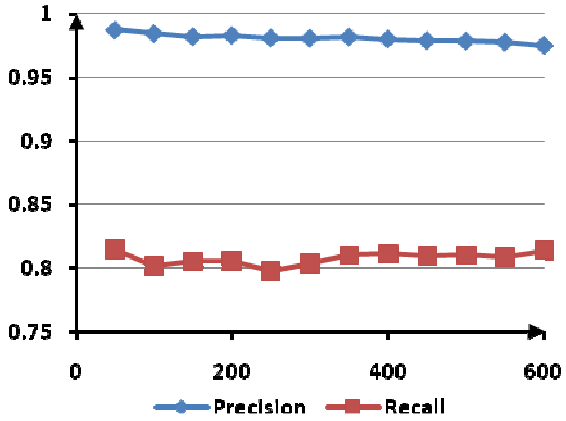

Figure 5. Performance on Amazon

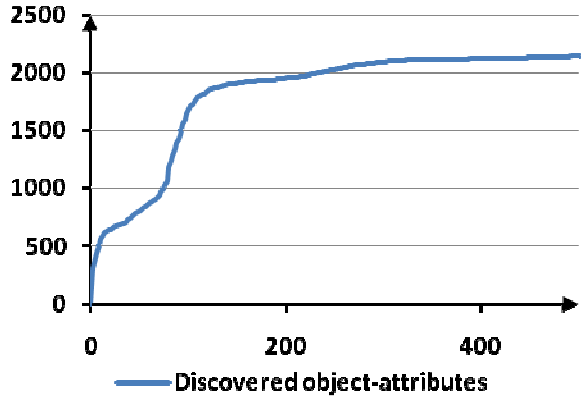

Figure 6. Object-attributes on different page scares of Zappos

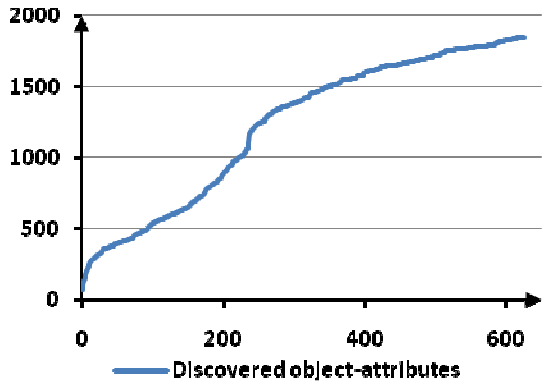

Figure 7. Object-attributes on different page scares of Amazon

In Table 4 we show the precision and recall of results derived by our approach with different clustering distance thresholds. The dataset we use contains about 10 thousand pages from Zappos and 10 thousand pages from Amazon. Performance on the entire Zappos and Amazon data sets are presented in Table 5.

Our experiments show that the clustering distance threshold, which is the only parameter in our approach, is easy to fix. Recall that in our clustering algorithm, if two pages belong to the same query, the only possible links which don't co-cite the same result pages are pagination links. Thus, the distance threshold is the number of different pagination links between two pages. As we known, each paged Web page has its own serial number. Nearly all Websites design their pagination hyperlinks in the slide window way. That is, result pages for same query are sorted by their serial number and each result page presents hyperlinks point to *n* neighboring result page. So it is easy to fix this parameter by investigating how many *n* is. For example, Zappos.com is 4 and Amazon.com is 5. Table 4 shows the performance of our approach on about 10 thousands of pages with various clustering distance. The distance parameter affects the performance evidently that the precision drops but the recall grows with the threshold increases. When the threshold equals to the experiential value, the precision and recall reach its peek value (shown in bold font) at the same time. Thus in practice, we can fix this parameter preliminarily by user experience and then validate it on small data set. All of our succeeding experiments fix their clustering distance for Zappos and Amazon respectively as 4 and 5. The excellent results on various data scale argue that clustering distance parameter affects the performance stably.

In Figure 4 we show the online precision and recall as our approach produce results on the crawled dataset. The horizontal axis shows the total number of objects (in thousands) the method has produced so far. We carry out the same test for Amazon data, and result is shown in Figure 5. Figure 6 and Figure 7 show the number of object-attribute pairs we recover given the number of web pages (in thousands) we have processed. Since we have the entire dataset of Zappos, the number of additional new pairs reduces as we process more and more pages. This phenomena is not obvious for the Amazon dataset, as we work on a subset of Amazon data.

The performance of our approach is good. No matter it is on the entire Website (Zappos) or on an incomplete Website (Amazon), the precisions are as high as 0.98. And the recall of results from Zappos is also good. The recall of results from Amazon is a little lower than that from Zappos.

Interestingly, the recall of our approach depends not only on the completeness of the dataset, but also on the speed of crawling.

The reason why recall on Zappos, as shown in Table 5, cannot achieve 100% is because the Website updates their products during our crawling process. So some pages in our data sets are inconsistent. For example, the product "Tunnel Vision Shirt" for query "clothing & kids" existed on Zappos.com when we crawl the result pages of query "clothing & kids", but it was removed when we crawl result pages of query "clothing", then our approach will not identify hyperlinks on pages of "clothing" to pages of "clothing & kids" as drill-down links. So "clothing" will not be labeled to "Tunnel Vision Shirt". The reason why the recall of results from Amazon is lower than that from Zappos is that the Amazon data set lacks some necessary Web pages because the dataset we have is not complete. Definitely, if some result pages for a query are not crawled, the objects on these result pages will not be retrieved. More important, the object set for the query is incomplete that leads to the drill-down link detection for those result pages may fail. Nevertheless, the recall still reaches 0.80.

Table 4. Performance on a small data set with different thresholds

Threshold	Zappos		Amazon	
	Precision	Recall	Precision	Recall
2	0.97	0.84	1.0	0.58
3	0.97	0.90	0.99	0.77
4	**0.97**	**0.96**	0.99	0.80
5	0.82	0.96	**0.99**	**0.84**
6	0.64	0.96	0.39	0.84

Table 5. Performance on total data set

Data Set	Discovered object-attributes	Precision	Recall
Zappos	2155059	0.98	0.97
Amazon	1853145	0.98	0.80

The factor affects precision is that the result Pages detected by page type identification algorithm contain some fake result page. Some unrelated pages contain links to recommended product pages. If the object set of an unrelated page, for example "Brands" taxonomy page, happens to be subset of the other object set which derives from a query, then hyperlink to the "Brand" taxonomy page will be detected as useful query link. As a result, our algorithm will label "Brands" to the recommendation products. However, "Brands" is not a meaningful attribute value, but an attribute name. So the results on entire Website pages can't achieve 100%. But the number of taxonomy pages is quite smaller than result pages. Thus, the number of recommended products is small. And many hypertexts of taxonomy page hyperlinks make sense, such as "Women", "Beauty" on Zappos and "Books", "Electronics" on Amazon.

The performance of our approach is stable. As shown in Figure 4 and Figure 5, the precision and recall are stable, don't warp over 0.01, on various object sets whose sizes are from 20 thousands to 600 thousands.

The main cost of our approach is to check Web pages and our goal is to discover object-attribute pairs. So we estimate the overhead by checking the how many object-attributes can be discovered on different numbers of Web pages.

On complete data set, Zappos, our approach can discover most of the object-attributes fast. As we can see in Figure 6, the number of discovered object-attributes grows rapidly with the number of scanned Web pages increases at the beginning, and about 84% object-attributes are discovered when 110 thousands of Web pages are checked. But the curve grows slowly after the number of scanned Web pages larger than 110 thousands. This is because most of the object-attributes contained by the last Web pages have been discovered in the previous Web pages. This result valid our assumption that lots of Web pages contain repeated information.

On incomplete data set, Amazon, our approach can discover object-attributes at a stead speed, as shown in Figure 7. Although many Web pages are missed in Amazon data set, the number of object-attributes discovered by our approach linearly relates to the scale of scanned Web pages.

6. CONCLUSION AND FUTURE WORK

In this paper, we build an interaction model named Exploratory Search Model and propose a unsupervised approach to discover attributes, especially hidden attributes doesn't exist on static Web pages from links among Web pages. The main foundation of our work as presented in the built model is the semantic embedded in link structure of exploratory Websites. In our unsupervised approach, the hidden attributes contained in hyperlinks, which can't be extracted from static pages using existing technologies, can be discovered automatically without any template; only a clustering threshold and a pattern for detecting object page need to set. Our experiments on large data sets, one of which is complete and the other one of which is incomplete, validate the effectiveness, general, efficiency of our approach.

However, there are still much works to do in future. Firstly, even if we can crawl entire Website, there are too many repeated information in the whole page set that a lot of Web pages need to check after 84% object-attributes are discovered. Secondly, it is hard to crawl entire Website in fact that will decrease the recall. Thirdly, the Websites may maintain their objects so as to decrease the recall. The approach to cover above problems is combine our unsupervised approach and crawling technologies to implement a online system, and develop more heuristic methods in order to avoid checking repeated information.

7. ACKNOWLEDGMENTS

The authors are grateful to MSRA and NUDT for offering experimental equipments and directions.

8. REFERENCES

[1] Buttler, D., Liu, L. and Pu, C. A Fully Automated Object Extraction System for the World Wide Web. In *Proceedings of the The 21st International Conference on Distributed Computing Systems* (2001). IEEE Computer Society.

[2] Ghani, R., Probst, K., Liu, Y., Krema, M. and Fano, A. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8, 1 2006), 41-48.

[3] Wang, J., Shao, B., Wang, H. and Zhu, K. Understanding Tables on the Web. *Under submission* (2010). Tech. Report.

[4] Probst, K., Ghani, R., Krema, M., Fano, A. and Liu, Y. *Extracting and Using Attribute-Value Pairs from Product Descriptions on the Web*. Springer-Verlag, City, 2007.

[5] Zhu, J., Nie, Z., Wen, J.-R., Zhang, B. and Ma, W.-Y. Simultaneous record detection and attribute labeling in web data extraction. In *Proceedings of the 12th ACM SIGKDD* (Philadelphia, PA, USA, 2006). ACM.

[6] English, J., Hearst, M., Sinha, R., Swearingen, K. and Yee, K.-P. Hierarchical faceted metadata in site search interfaces. In *Proceedings of the CHI '02 extended abstracts on Human factors in computing systems* (Minneapolis, Minnesota, USA, 2002). ACM.

[7] Marchionini, G. Exploratory search: from finding to understanding. *Communications of the ACM*, 49, 4 2006), 46.

[8] Fu, W.-T. The microstructures of social tagging: a rational model. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work* (San Diego, CA, USA, 2008). ACM.

[9] Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P. DOM-based content extraction of HTML documents. In *Proceedings of the Proceedings of the 12th international conference on World Wide Web* (Budapest, Hungary, 2003). ACM.

[10] Wang, F., Li, J. and Homayounfar, H. A space efficient XML DOM parser. *Data & Knowledge Engineering*, 60, 1 2007), 185-207.

[11] Wood, L. Programming the Web: the W3C DOM specification. *IEEE Internet Computing*, 3, 1 1999), 48-54.

[12] Choi, B. and Yao, Z. Web Page Classification*. *Foundations and Advances in Data Mining*2005), 221-274.

[13] Cooley, R., Mobasher, B. and Srivastava, J. Data preparation for mining world wide web browsing patterns. *Knowledge and Information systems*, 1, 1 1999), 5-32.

[14] Abiteboul, S. Querying Semi-Structured Data. In *Proceedings of the 6th International Conference on Database Theory* (1997). Springer-Verlag.

[15] Maynard, D., Tablan, V., Ursu, C., Cunningham, H. and Wilks, Y. *Named entity recognition from diverse text types*. Citeseer, City, 2001.

[16] Lehnert, W., McCarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F., Dolan, C. and Goldman, S. UMass/Hughes: description of the CIRCUS system used for Tipster text. In *Proceedings of TIPSTER' 93 workshop* (Fredericksburg, Virginia, 1993). ACL.

[17] Muslea, I., Minton, S. and Knoblock, C. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4, 1 2001), 93-114.

[18] Kushmerick, N. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118, 1-2 2000), 15-68.

[19] Mansuri, I. R. and Sarawagi, S. Integrating Unstructured Data into Relational Databases. In *Proceedings of the 22nd International Conference on Data Engineering* (2006). IEEE Computer Society.

[20] Seymore, K., McCallum, A. and Rosenfeld, R. *Learning hidden Markov model structure for information extraction*. City, 1999.

[21] Holzinger, W., Krpl, B. and Herzog, M. Using ontologies for extracting product features from web pages. *The Semantic Web-ISWC 2006*2006), 286-299.

[22] Zhu, J., Nie, Z., Wen, J.-R., Zhang, B. and Ma, W.-Y. 2D Conditional Random Fields for Web information extraction. In *Proceedings of the 22nd international conference on Machine learning* (Bonn, Germany, 2005). ACM.

[23] Han, X. and Zhao, J. *CASIANED: People Attribute Extraction based on Information Extraction*. City, 2009.

[24] Zhai, Y. and Liu, B. Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web* (Chiba, Japan, 2005). ACM.

[25] Gatterbauer, W., Bohunsky, P., Herzog, M., Kr\, B., \#252, pl and Pollak, B. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web* (Banff, Alberta, Canada, 2007). ACM.

[26] Lerman, K., Getoor, L., Minton, S. and Knoblock, C. Using the structure of Websites for automatic segmentation of tables. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data* (Paris, France, 2004). ACM.

[27] Doorenbos, R. B., Etzioni, O. and Weld, D. S. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the first international conference on Autonomous agents* (Marina del Rey, California, United States, 1997). ACM.

[28] Teo, T. and Choo, W. Assessing the impact of using the Internet for competitive intelligence. *Information & Management*, 39, 1 2001), 67-83.

[29] Artiles, J., Gonzalo, J. and Sekine, S. *Weps 2 evaluation campaign: overview of the web people search clustering task*. City, 2009.

[30] Wu, W., Li, H., Wang, H. and Zhu, Q. Towards a Universal Taxonomy of Many Concepts. *Under submission* (2010). Tech. Report .