Query Routing: Finding Ways in the Maze of the Deep Web

Govind Kabra Chengkai Li Kevin Chen-Chuan Chang
Department of Computer Science, University of Illinois at Urbana-Champaign
gkabra2@uiuc.edu, cli@uiuc.edu, kcchang@cs.uiuc.edu

Abstract

This paper presents a source selection system based on attribute co-occurrence framework for ranking and selecting Deep Web sources that provide information relevant to users requirement. Given the huge number of heterogeneous Deep Web data sources, the end users may not know the sources that can satisfy their information needs. Selecting and ranking sources in relevance to the user requirements is challenging. Our system finds appropriate sources for such users by allowing them to input just an imprecise initial query. As a key insight, we observe that the semantics and relationships between deep Web sources are self-revealing *through their query interfaces, and in essence, through the* co-occurrences *between attributes. Based on this insight, we design a co-occurrence based attribute graph for capturing the relevances of attributes, and using them in ranking of sources in the order of relevance to user's requirement. Further, we present an iterative algorithm that realizes our model. Our preliminary evaluation on real-world sources demonstrates the effectiveness of our approach.*

1. Introduction

The Web has been rapidly "deepened" by the massive *Web data-sources*: While the *surface Web* has linked billions of HTML pages, a far more significant amount of information is hidden in the *deep Web*, behind query forms of the Web data-sources like *amazon.com*. These Web data-sources are often also referred to as *hidden* or *invisible* Web. The July 2000 survey [3] claims that there are 500 billion hidden pages in 10^5 Web data-sources. Another study [5] estimates 450,000 such Web data-sources. Recently, attempts towards integrating such large scale deep Web data-sources has gained much attention. We discuss some of the research efforts related to our paper in Section 6.

Information underlying numerous such Web data-sources cannot be accessed directly through static URL links. These sources allow the users to access the underlying information by querying through their query interfaces. With myriad data-sources on the Web, a user may not know the sources that can satisfy her information requirements. Consider, user Amy in New York who is joining the graduate school in Chicago. What are the data-sources that can book her flight from New York to Chicago (e.g., *orbitz.com*)? Where can she purchase books for her graduate studies (e.g., *barnesandnoble.com*)? Where can she find a part-time job (e.g., *monster.com*)? In the deep-Web there may be many sources that can satisfy her information needs, but she may not be aware of their existence. This paper presents a system that provides a solution to this important problem: While the number of Web data-sources are in the order of 10^5, how to find the data-sources that are most relevant to the user's requirements?

Finding the relevant sources is challenging because the user's information requirement may not be distinctively clear. In our earlier example, Amy is interested in buying books for her graduate studies, but it is not distinctively clear whether she is interested in graduate books, or science books, or literature, or publications, or journals. The information requirements of users cannot be enumerated. We shall refer to users information requirement as her *domain of interest*.

Different users may be interested in different domains, thereby making the notion of domain very ad-hoc in nature. Further, the information that a user can access naturally depends on what is available on the deep web, and thus is dynamic. Thus, the domain of interest is ad-hoc, dynamic and not distinctively enumerated, and thus finding sources relevant to users domain of interest becomes very challenging.

However, observe that the query interfaces of the data-sources contain the attributes that tend to describe the information accessible through them. For example, the query interface of a source like *barnesandnoble.com* contains attributes such as *author, title, ISBN, etc.*, indicating that it is about books. The query interface of *orbitz.com* contains attributes such as *from, to, departure date*. Hence, even though the domain of interest may not always be distinctively clear, user would know certain attributes that tend to describe the object of her interest and therefore, are also likely to be used on the query interface of a relevant datasource. Capitalizing on this observation, we attempt to build a **source selection system**, where user may specify a set of attributes as input, and the system finds the sources that are most relevant to the information requirements of user. For instance, in earlier example, Amy can give an input query: *(from, to, departure date)* to find sources that allow her to book airline tickets.

The system we propose in this paper is robust to the incompleteness or impreciseness in user queries. This requirement becomes necessary with the observation that the users may not be aware of all the attributes, or even the best attributes for their domain of interest. In earlier example, for the query given by Amy, there are many other attributes, such as *cabin, airline, number of passengers, trip type* that are also relevant, but not specified in her query. Under such situation, the user query shall be incomplete and imprecise. As we shall see later, our system is able to find relevant answers even under such situations, and thus frees the user from the worries of choosing the best set of attributes to query our system.

The key insight, underlying the solution we develop, is that though the autonomous heterogeneous sources are seemingly independent, the information they provide is *"self-revealing"* through the attributes used in their query interfaces. To be more specific, we observe following two mutually recursive phenomenons:

- **Relevant attributes occur in relevant sources:** The attributes that occur often in sources relevant to some domain are more relevant to that domain.
- **Relevant sources contain relevant attributes:** The relevance of a source to some domain is reflected by the relevance of attributes used in its query interface.

As a consequence of above phenomenons, the attributes that are relevant to some domain co-occur with other attributes relevant to that domain very often. In other words, if some attributes are known to be relevant to a domain then an attribute that co-occurs often with these attributes is also likely to be relevant.

Based on this observation, our system constructs a co-occurrence based attribute linkage graph. Using this graph, and the user query, it finds the relevance of other attributes. Using the attribute relevances thus obtained, the system then finds the relevance scores for sources.

The co-occurrence based attribute relevance modeling is based on recursive observations and thus develops a *recursive reinforcing* relation for estimating attribute relevances. To make this model usable in an online system, we develop an iterative algorithm that can compute the attribute relevances on-the-fly, and thus rank the sources in order of relevance to users interest.

In IR community, models have been developed to find the Page-Rank [4] or hub-authority scores [13] of web pages using the hyperlink based web graph. However, these models are not directly applicable to our problem setting. In Section 2, we contrast these models with the underlying model of our system.

Many recent research efforts in the direction of large scale integration [9, 17, 10, 16] focus on "domain-based" integration issues, i.e., integrating the sources that provide access to similar type of objects. As an input to their system, these solutions require set of sources that are all about same domain. Clearly, such set of sources are not readily available, and requires tiring manual collection. In this regards, the methodology presented in this paper can be used to automate the process of generating input for such domain-based information integration solutions.

In sum, the main contributions of this paper are:

- We propose a novel **attribute co-occurrence graph** for modeling the relevance of attributes.
- We present an **iterative algorithm** that computes attribute relevances given just an **imprecise user query**, and thus find the relevant sources.
- We report **experiments based on dataset of real Web sources**. Our preliminary results demonstrate the effectiveness of our approach.

The remainder of the paper is organized as follows: In Section 2 we present the co-occurrence based attribute linkage graph, develop a framework to compute the attribute relevances and explain how to rank the sources using the attribute relevances. In Section 3, we give the details of our iterative solution that computes the attribute relevances, and thus ranks sources. A preliminary experimental evaluation of our approach is given in Section 4. We summarize the future research directions that we intend to pursue in Section

5. Finally, we discuss research efforts related to our work in Section 6, before we conclude our paper in Section 7.

2. Foundation: Attribute and Source Relevance

We begin this section with a formal specification of our problem. We then present the attribute co-occurrence based linkage graph. Next, we describe how we model the relevance of attributes to the domain of interest to user. Finally, we explain how we use these relevance to rank sources.

2.1. Problem Specification

Given the set of web data-sources and a user query containing a set of attributes, our goal is to find the set of sources that are most likely to satisfy the user's information requirements. Below we give some notations that we use throughout this paper.

Definition 1 (Web DataSources): We apply our algorithm on a set of data-sources $\{ds_1, ..., ds_m\}$. Each ds_i has an associated set of attributes occurring on its query interface denoted by S_i. The attribute vocabulary set, $\mathcal{A}$, is defined as union of the set of attributes on each of these data-sources, i.e., $\mathcal{A} = \cup S_i$. ■

Definition 2 (Source Ranking): Given web data-sources and a user query $\mathcal{V} = \{a_{v_1}, ..., a_{v_k}\}$, the goal is to find a source ranking, $\mathcal{R}$= $(ds'_1, ..., ds'_m)$, in the order of relevance to $\mathcal{V}$. ■

2.2. Attribute Co-Occurrence Graph

Every data source has a query interface containing a set of attributes. A user can access the information underlying these data sources by specifying values for these attributes. These attributes tend to give an indication of the type of information accessible through the query interfaces they occur in.

Consequently, the attribute sets corresponding to the query interfaces of different sources providing similar information are likely to have many common attributes. For example, consider the query interfaces at the data sources - amazon.com and barnesandnoble.com providing information about books. Both of them contain the attributes - *author, title, publisher, ISBN*. However, one should carefully note that all the sources that provide similar information do not have exactly the same set of attributes on their query interfaces. More specifically, there may be some attributes in $\mathcal{A}$ that occur in large number of such data sources, while there may be some other attributes that occur in fewer of these data sources, and rest of the attributes not occurring at all in any of these sources. An attribute that is more frequent across the query interfaces of sources providing similar information can be expected to be more important, i.e., a source where such attribute occurs is more likely to provide the similar information. In order to quantify this importance measure, we associate a *relevance score* with each attribute. This relevance score represents how likely a query interface containing this attribute will provide information of interest to user.

We develop a methodology wherein relevance of attributes is determined using the relevances of other attributes. We discuss in more detail about this in Section 2.3. In this section we now describe the details of our co-occurrence based attribute graph.

In this graph view, there is a vertex (also referred to as node) corresponding to each attribute in the attribute vocabulary. Two attributes are said to co-occur if they are both present in some source. The attribute co-occurrence graph has weighted directed edges between any two nodes corresponding to some co-occurring attributes. We detail how to determine the weights for these edges shortly. Before that, we would like to draw attention of our readers to the following observation.

A graph constructed as above shares some similarity with the hyper-link based web graph used by search engines (e.g., google.com). The nodes in two graphs correspond to attributes versus web pages. The edges are directed in both the graphs. The weights of these edges in attribute graph model is based on the number of sources the two attributes co-occur in, while in the web graph the weights are based on the hyper-links between pages. Such a web graph is used by search engines to determine the importance(or page rank) of different web-pages. We develop similar framework to determine the relevance score for each of the attributes.

Inspite of various similarities that our approach shares with page rank modeling, the two approaches have a sharply distinguishing aspect. The page rank is concerned about finding the static importance of every page, while in our case, we determine the relevance of an attribute for every input query. More specifically, we do not have any notion of static importance of an attribute. For every user query, we apply our framework to determine the relevance of each attribute to the users interest.

Now we describe how the edge weights are determined in co-occurrence based attribute graph. Then we move on to discuss how we use this graph to model attribute relevances.

We denote the weight of the directed edge $(a_i \rightarrow a_j)$, between the nodes corresponding to attributes a_i and a_j as w_{ij}. This weight gives the conditional relevance measure of attribute a_j given the attribute a_i is known to be relevant. The idea here is that more frequently the attribute a_j co-occurs with a_i as compared to other attributes co-occurring with a_i, higher is the degree to which it gets the relevance induced from a_i. Thus, this weight gives the degree to which relevance is induced from a_i to a_j. Mathematically, if $countSources(a_i \wedge a_j)$ denote the number of sources that attributes a_i and a_j co-occur in, then w_{ij} is given by following expression.

$$w_{ij} = \frac{countSources(a_i \wedge a_j)}{\sum_{a_k \in A} countSources(a_i \wedge a_k)} \quad (1)$$

2.3. Attribute Relevance Modeling

The user query contains a set of key attributes that the user thinks will be relevant to her information needs. This means that these are the set of attributes that are likely to be used in query interfaces of the sources that provide information that is of interest to user.

As motivated earlier, the user query is both incomplete and imprecise. It is incomplete because there may be some attributes that are not specified in the input query, but may occur in the query interfaces of sources of interest to user. It is imprecise because the key attributes do not necessarily have to occur in the query interfaces of sources of interest to user. As we mentioned in Section 2.2, we associate with each attribute a relevance measure. This relevance measure indicates how likely the attribute is to be used in a query interface of interest to user. In this section we describe how to model these relevance measures.

The attributes that are specified in the user query have strong relevance to users interest. There may be some other attributes that are not present in the user query, but occur in many of the data sources in which the input attributes occur. There is a strong likelihood that these attributes that co-occur with input attributes are also relevant. Furthermore, the attributes that co-occur with these set of attributes are also likely to have some relevance to user query, and so on.

Essentially, any relevant attribute shall lead to increasing the relevance of its co-occurring attributes. Such propagation of relevances may take place over multiple hops of co-occurrences, thereby resulting in each attribute having some relevance to user query.

In order to discriminate the attributes on the basis of their degree of relevances, we need to carefully model such relevance propagation. We use a model similar to the page rank model, where some important page induces all the pages to which it has hyperlinks in proportion to the strength of the outward hyperlink based edges. In our case also, an attribute influences each of its co-occurring attribute in proportion to the weights of the outgoing edges it has to those attributes. The relevance of an attribute is determined by aggregation of the relevances induced by each of its co-occurring attributes. For the purpose of this paper, we use an additive scheme to aggregate the relevances coming in via different incident edges, and leave the study of other sophisticated combination functions to future study.

Let us denote the relevance of an attribute a_j by $P(a_j)$, then the formulation we obtain is as follows:

$$P(a_j) = \sum_{a_i \in A \setminus a_j} d.w_{ij}.P(a_i) \quad (2)$$

The decay factor d in this formulation implies that when a node influences its co-occurring attributes, it does this only by a fraction d of its own relevance measure. This has correspondence to the Page-Rank model where from the current page the random surfer takes one of the outgoing links with some probability, d, and jumps to any arbitrary page with probability $(1 - d)$. Such a decay factor is important in our model because there could be cycles in our graph, and the relevances may keep propagating infinitely along these cycles.

It is important to note here that these attribute relevance measures are not probabilities. But they give us a relative measure to discriminate between different attributes with regards to their degree of relevances.

Notice that the formulation in Equation 2 involves all the attributes in $\mathcal{A}$, which may contain attributes that never co-occur with a_j. This broad inclusion is not inconsistent with our development because for such attributes the edge weight, w_{ij}, will be zero.

2.4. Source Relevance

Now we describe how to use the attribute relevance formulated as described above to find the relevance score for different sources. Every data source has a set of attributes on its query interface. The relevances of attributes give a measure of how likely the interfaces on which they occur are of interest to user. Consequently, the sources that contain more attributes with higher relevances are going to have higher likelihood of being relevant to users information requirements. If a source contains, many attributes with lower relevance, it is likely that the information provided by such a source is not of interest to user. Therefore, the source relevance score formulation should give higher score to the sources that contain more number of high relevance attributes and less number of low relevance attributes. Mathematically, the score for a data source ds_i is denoted as $score(ds_i)$, and is given by:

$$score(ds_i) = \frac{\sum_{a \in S_i} P(a)}{|S_i|} \tag{3}$$

In the above formula, normalization with $|S_i|$, i.e., the number of attributes on the query interface of ds_i, is necessary as this would penalize the sources having attributes with lower relevances. Thus using this function assigns high score to sources that contain more number of high relevance attributes, and lesser number of low relevance attributes.

The above ranking function is very simple in nature. There is a wealth of ranking functions that have been studied in IR research. In order not to lose our focus, we leave investigation of other scoring functions for future explorations.

However, we would like to draw the attention of our readers to the contrast between above scoring function and TFIDF, a very popular scoring measure in IR domain. Firstly, in order to capture the importance of any key-word, TFIDF penalizes it by its frequency across the documents, while in our solution, we have the estimates of the relevance of each attribute readily available. Secondly, we want to penalize the sources where some attributes are not very relevant, and realize this using normalization with $|S_i|$.

From the discussion so far, the above two step procedure for finding relevant sources seem to be complete. But if we observe Equation 2 carefully, we see that it is a *recursive reinforcement* formulation where estimation of relevance value for a_j requires these estimates for all attributes co-occurring with a_j. For a non-trivial attribute co-occurrence graphs, the relevance estimates for all co-occurring attributes may not be available when we have to compute the relevance of a_j. The complexity of this problem becomes more evident with the realization that the attributes co-occurring with a_j cannot themselves estimate their own relevances unless the relevance of a_j is known.

Similar *recursive* formulations have also been encountered for computing *Page Rank* of a page, which depends on the *Page-Rank* of other linked pages. In such situations, iterative solutions have been developed that eventually converge to the correct estimates.

As we already mentioned earlier, that *Page Rank* needs to be computed in an off-line fashion, therefore efficiency issues are not as critical to the solutions developed. In contrast, in our problem setting, we need to run our solution in an online fashion for every user query.

This requirement makes the designs of iterative solutions for page rank computation unsuitable in this scenario. In next section, we present an alternative iterative algorithm that allows for attribute relevances to be computed in an online fashion. Note, however, that the source ranking procedure using the attribute relevances thus obtained remains the same as discussed above.

3. Online Algorithm: Iterative and Progressive

We now present an algorithm that iteratively estimates the attribute relevance values. The general idea is that the relevance score of some attribute is re-estimated at each iteration, thereby triggering changes in relevance scores of some other attributes. These affected attributes are then queued up for updation of their relevances in future iterations.

We begin this section by giving a very simplified view of the estimation mechanism. Suppose at some iteration, attribute relevance is β for a_j and is α_i for rest of the attributes a_i. From Equation 2, we know that $\beta = \sum_{a_i \in A \setminus a_j} d.w_{ij}.\alpha_i$. Now suppose at some iteration, $P(a_1)$ changes from α_1 to α_1', resulting in $P(a_j)$ to change from β to β^*, such that, $\beta^* = w_{1j}\alpha_1' + \sum_{a_i \in A \setminus a_j, a_1} d.w_{ij}.\alpha_i$.

Instead of updating β to β^* as above, we alternatively update $P(a_j)$ using the following mechanism. We show that the estimate for $P(a_j)$ obtained thus is also β^*.

To begin with, we calculate δ_{1j}, a quantity that would be used to adjust $P(a_j)$ to account for the change in $P(a_1)$.

$$\delta_{1j} = d.w_{1j}.(\alpha_1' - \alpha_1) \tag{4}$$

Now we apply δ_{1j} on to the old estimate of $P(a_j)$, β, to give us its new estimate, β', as follows:

$$\begin{aligned} \beta' &= \delta_{1j} + \beta \\ &= d.w_{1j}.(\alpha_1' - \alpha_1) + \sum_{a_i \in A \setminus a_j} d.w_{ij}.\alpha_i \\ &= d.w_{1j}.\alpha_1' + \sum_{a_i \in A \setminus a_j, a_1} d.w_{ij}\alpha_i \end{aligned}$$

Thus we see that estimate, β', obtained above is exactly the same as β^*.

Informally speaking, when $P(a_1)$ changes from α_1 to α'_1, node a_1 informs node a_j about this change by sending it a quantity δ_{1j}. Node a_j then applies δ_{1j} on its current relevance estimate, β, to adjust it to a new value β'. This alternative mechanism forms the basis for our algorithm.

At each iteration, our algorithm adjusts the relevance score of one of the attributes that has some pending δ. We call this procedure as expanding the attribute node. In general situation, we may not be able to expand a_j immediately following the expansion of a_1. This may happen if we choose to expand some other attribute nodes for next several iterations. During these iterations, the relevance scores of some more co-occurring attributes of a_j may change, and they would send appropriate δ value to node a_j. Moreover, we could have chosen to expand some co-occurring node of a_j more than once during these iterations, which means there could be more than one δ value pending from any of the co-occurring node of a_j. All the δ values transferred to a_j need to be memorized until the iteration in which we expand a_j.

For such memorization, we maintain a quantity Δ^u with every attribute a_u, which is set to zero whenever a_u is expanded. While expanding a_u, Δ^u is applied on its current relevance score, α_u, thereby changing it to α'_u, such that $\alpha'_u = \Delta^u + \alpha_u$

Corresponding to this change, a_u transfers some $\delta_{u,v}$ to each of its co-occurring attribute a_v, such that $\delta_{u,v} = d.w_{u,v}.(\alpha'_u - \alpha_u)$.

Each of these co-occurring attributes aggregate incoming $\delta_{u,v}$ with their Δ^v values such that $\Delta^v = \Delta^v + \delta_{u,v}$.

Thus, every time node a_u receives some δ resulting from expansion of one of its co-occurring attribute, it updates Δ^u value. Eventually, at some iteration when a_u is expanded, Δ^u is applied to its most recent estimate for relevance score to obtain a new estimate.

Having informally discussed our iterative algorithm, we give a high-level description of the algorithm in Figure 1.

Higher the value of Δ, higher is the change in relevance score. Consequently, higher are the values of δ transferred to co-occurring nodes. Thus, the higher value of Δ are expected to trigger larger changes. We, therefore, prioritize our node expansion in the order of pending Δ.

Such careful prioritization implies that the nodes expecting higher changes in relevance values will be expanded earlier. In contrast, the earlier work on Page Rank computation estimates the page rank of every page at every iteration. In our system, there is no notion of static relevance measure, but the relevance measure here needs to be computed for every user query. Our iterative mechanism allows us to iterate on those attributes whose relevance values are likely to undergo higher changes. Such methodology is, therefore, more suited to our problem setting.

```
Input:
  • $\mathcal{V}$: input query attributes.
Output:
  • $\mathcal{R}$: ranked list of sources
Procedure:
01. $\forall a_i \in \mathcal{V}$, $\Delta^i \leftarrow 1$; UPDATEORADD($\mathcal{Q}, a_i, \Delta^i$)
02. while $\mathcal{Q}$ is not empty and limit not reached
03.     $(a_u, \Delta^u) \leftarrow$ POP($\mathcal{Q}$)
04.     $\alpha_u \leftarrow$ P[u]
05.     $\alpha'_u \leftarrow \Delta^u + \alpha_u$
06.     for each $a_v \in adjacent[a_u]$
07.         $(a_v, \Delta^v) \leftarrow$ PEEP($\mathcal{Q}, v$)
                //if it doesn't exist set $\Delta^v$ to 0
08.         $\delta_{u,v} \leftarrow d.w_{u,v}.(\alpha'_u - \alpha_u)$
09.         $\Delta^v \leftarrow \Delta^v + \delta_{u,v}$
10.         UPDATEORADD($\mathcal{Q}, a_v, \Delta^v$)
11.     P[u] $\leftarrow \alpha'_u$
12. $\mathcal{R} \leftarrow$ RANKSOURCES($P$)
13. return $\mathcal{R}$
```

Figure 1. Online Iterative Algorithm

More recent works on developing algorithms for faster page rank computation essentially iterate more on those set of pages that are expected to have higher changes in their page ranks. We believe our solution can also be extended for faster page rank computation and thus offers an alternative to these recent developments. Further investigation on this is beyond the scope of this paper, and have been left to future studies.

4. Experimental Evaluation

We begin this section by presenting the experimental setup. Then we present results of several experiments, and also give insight into how our algorithm is able to retrieve good quality results.

We manually collected a dataset of Deep Web Sources using Web Directories(e.g., *invisibleweb.com, brightplanet.com, webfile.com*) and Web search engines (e.g., *google.com*). The dataset contains 494 Web query interfaces and totally 370 attributes providing information about diverse domains, viz., airfares, automo-

Query 1	(from, to, departure date, return date)
Query 2	(author, title, ISBN)
Query 3	(make, model, price)
Query 4	(song, album, artist)
Query 5	(title, actor, director)
Query 6	(from, to)
Query 7	(from, to, adult, child)

Figure 2. Test Queries

biles, books, car rentals, hotels, jobs, movies, and music records. This dataset is part of the *UIUC Web Integration Repository* and available on-line for public viewing at http://eagle.cs.uiuc.edu/metaquerier.

We extracted the query interfaces of these data sources. These query interfaces are designed by Web programmers, who use some text that describe the attributes. We find that the text used to describe the same attributes on different query interfaces may differ to various degrees, ranging from some minor syntactic variations, to the use of synonymous words. The dataset we use in following experiments has been obtained by manually applying simple cleaning rules on the original dataset.

To evaluate the effectiveness of our algorithm we tested it over various queries listed in Figure 2. The algorithm takes as input the set of attributes in the user query. For instance, if a user is interested in buying airline tickets, then the input attributes of the query can be *(from, to, departure date, destination date*), as shown in *Query 1* of Figure 2.

4.1. Quality of Results

We run our algorithm for each of the input queries in Figure 2, iteratively until Δ values fall below 0.001. We set the value of d at 0.85. Upon termination, the algorithm has a relevance value associated with each attribute. We summarize the top-5 attributes for each of the test query in Figure 3(a). The algorithm then uses these attribute relevances to compute a relevance score for each data source. We summarize the top-30 sources returned by this algorithm in Figure 3(b).

The y-axes of the two figures correspond to the test Queries 1-7, while the x-axes has the id numbers of attributes (Figure 3(a)), and of data sources (Figure 3(b)). A point (x, y) in Figure 3(b) represents that the source with index x was in the top-30 data sources for *Query y*. Similarly in Figure 3(a), it would represent that the attribute with id number x was in the top-5 attributes for *Query y*.

Observe the results for *Query 1*, *Query 2* and *Query 3*. The set of top-5 attributes for these queries have small overlapping, and the top-30 sources are very different. This should not be surprising because these queries correspond to users who are interested in very different types of sources, viz., airlines, books and automobiles, respectively. Thus we see that our algorithm is able to identify the sources that best correspond to domain the user is interested in and does not output sources that are not related to that domain.

While on the other hand, *Query 4* and *Query 5* have many common top-5 attributes; Also, they have some top-30 sources that are different, while a set of sources are present in top-30 results for both the queries. These queries are for e-commerce Web sites for Music Records and Movies. In the real Web, there are some sources that only sell either Music Records, or Movie CD. While there are many Web sources that would give access to both types of contents, and require the user to specify the values for attributes that are related to both domains. Using this algorithm we are indeed able to return both categories of sources.

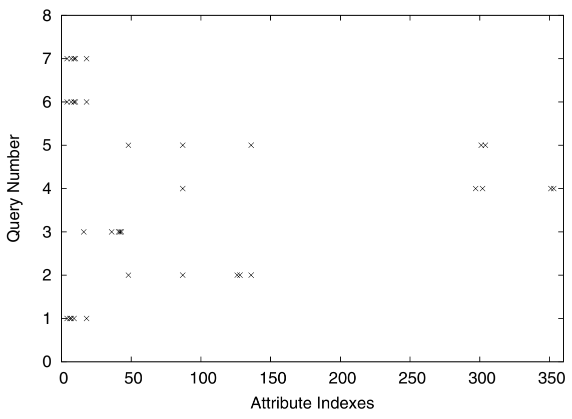
Robustness to imprecise and incomplete queries

We now demonstrate a very appealing property of our system. Observe that, *Query 1*, *Query 6*, and *Query 7* are all likely to be posed by a user who is interested in e-commerce Web sites in airfare domain. These queries differ in terms of the input attributes. This is likely to happen very often, because users of our system may not always know the best set of attributes for the domains they are interested in. Observe the top ranked attributes and the top ranked sources obtained for these 3 queries in Figure 3(a) and Figure 3(b) respectively. As we can see, the results obtained are very similar disregarding the variation in the input query attributes.

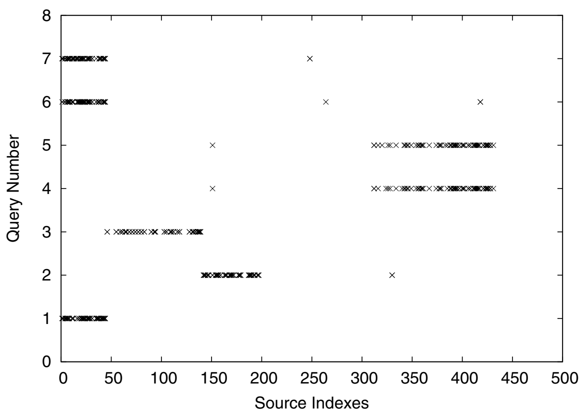
This property of our system frees the user to worry about finding the right set of attributes to query with. Even with a set of attributes that may not best describe the type of sources she is interested in, the algorithm will be able to direct her to the right set of sources.

Zooming into the Graph: Finding other relevant attributes

In this experiment, we study the attributes with high relevance. In interest of space, we limit this discussion to *Query 1*. This query is likely to be posed by a user who is interested in finding some e-commerce Web sites that provide information about airfares. We summarize the top-15 attributes for this query in Figure 4.



(a) Top-5 attributes for each query



(b) Top-30 sources for each query

Figure 3. Evaluation over Test Queries

Rank	(Index)	Name
1	(6)	departure date
2	(7)	return date
3	(18)	to
4	(4)	from
5	(9)	adult
6	(10)	child
7	(16)	trip type
8	(8)	class
9	(19)	return
10	(12)	infant
11	(1)	city
12	(14)	departure
13	(15)	senior
14	(28)	airline
15	(2)	destination

Figure 4. Top15 relevant attributes for Query 1

The top-4 attributes in Figure 4 are contained in *Query 1*. Based on the co-occurrence analysis, the algorithm is able to zoom into one region of the attribute connectivity graph, and is able to find other attributes that are also relevant. In particular, attributes - *adult*, *child*, *infants*, *trip type*, *cabin* and *airline* are other attributes our algorithm predicts to have high likelihood of being relevant to airfare domain.

Robustness to Noisy data The texts used for same attribute across different query interfaces are often syntactic variations, or sometimes usage of synonymous words describing the same attributes. This renders our data set to be noisy. As we mentioned earlier, we applied simple cleaning rules to resolve straightforward synonymous words. Such simple rules are not able to identify that texts - *to*, *destination*, *destination city* and *to city* actually refer to the same attribute. But if we observe Figure 4 carefully, we see that it also assigns high relevance values to *return* which actually refers to *return date*; *departure* and *destination*, that are variations for attributes *from* and *to* respectively.

Thus our algorithm is robust to such noises in data, and can deliver good quality answers even in presence of such noises in data. From this perspective, our work can be viewed as the generalization of correlation mining based schema matching [9]. Their approach is limited in two ways. Firstly, their solution can be applied only when all the sources are about same object. Secondly, they use only immediate co-occurrence statistics. By successively propagating Δ values across edges of our graph, we are able to elegantly capture multi-hop-co-occurrence statistics, without requiring the sources to be partitioned into different domains.

5. Future Research Directions

Progressive Result Generation

In our experimental studies, we observed that our system is able to retrieve highly relevant sources. As we noted, that the higher values of Δ leads to higher changes in the preference values for attributes, and therefore priortizing our iterative algorithm over values of Δ enables us to find the relevant sources quicker.

The performance requirement of different users of our system shall vary. Some users may be interested only in retrieving the Top-10 sources for their input query, while some others may be interested in Top-30 sources. Finding all relevant sources may thus become unnecessary for every user. Builiding upon the same hypothesis, that higher values of Δ shall lead to larger changes in attribute relevances, we believe that we should be able to tune our system to generate the relevant sources progressively. More specifically, we believe that due to priortization based on Δ values, our system should be able to find Top-10 sources before it has found the later sources. Thus, we can tune our system to present to the user the Top-10 sources as soon as they are available, while continue to compute the sources ranked 11^{th} to 20^{th} in the background.

Stemming of Attributes

The user in our system is required to submit an exact matching attribute to the one in our repository. We believe such exact matching of attributes is undesirable. We therefore intend to split the text of every attribute name into its constituting words, and use those words for partial matches of the input queries. This if done in a naive way, would magnify the size and complexity of our attribute co-occurrence graph, and also may produce spurious results. We are currently working on developing sophisticated ways to implement this idea.

6. Related Work

A wealth of work has been done in general information integration systems [15, 7], where sources are configured *a priori* for specific tasks. Several integration systems have been developed in a relatively small scale: *e.g.* Information manifold, TSIMMIS, Infomaster, Garlic, DISCO, Softbot, Ariadne, and others. In contrast, for text databases, there have been more efforts in large scale distributed search, *i.e.*, meta-search (*e.g.* [8]).

Database sources on the Deep Web has gained much attention from the community recently. Ipeirotis et al. [12, 11] classify and summarize contents of text databases by sending probing queries to the sources. Raghavan et al. [14] have built a crawler for the deep web. The work of [6] and [1] discuss the wrapper generation techniques. He and Chang propose statistical approach for schema matching over deep Web sources [9]. Zhang et al. [17] propose a parsing framework for extracting Web query interfaces. He [10] and Wu et al. [16] apply clustering technique for integrating deep Web interfaces.

An extensive survey [5] on deep Web was conducted to observe characteristics of the sources and study the implications of these characteristics on exploring and integrating Web databases. It clearly identifies large-scale as a major challenge that exists for integration of structured databases.

Query expansion technique in IR and Web search expands users' original query with new terms and re-weights the terms in the expanded query [2]. The inherent idea of using co-occurrences of our probabilistic model shares similarity with these efforts. In our scenario, attribute vocabulary has been known to converge [5], therefore, it was possible for us to develop an extensive modeling of attribute relevance that cleanly subsumes these past efforts.

Recursive definitions are used to find the importances of Web pages [13, 4], where iterative solutions have been developed to make such formulations computable. Our approach share some similarities with these work. The iterative solution used in these work are not suitable for our problem setting. We propose more sophisticated iterative solution in this paper to find the relevance of attributes, and thus find the high ranked sources.

7. Conclusion

In this paper, we present a co-occurrence based attribute relevance model and an efficient iterative algorithm for finding Web sources relevant to users information requirements. Our preliminary experiments on real-world sources illustrate that sources can be ranked reasonably well to match users' interests. This paper thus opens an interesting research direction for the novel problem of source ranking on deep Web. Important issues to investigate in future and potential solutions with key insights have been identified.

Our solution for finding the right sources can be viewed as search problem in deep-web. A classic analogy comes from surface-web information retrieval domain, where we saw recent developments of keyword-based search engines (*e.g.*, Google) vs. early attempts of organizing web directories (*e.g.*Yahoo!). We do not require to draw hard boundaries or partition our repository into domains because the notion of domain in our study is only virtual. Given the input attributes, the algorithm harnesses the statistical information in data to identify the most relevant set of sources.

Our work can also be viewed as on-the-fly partitioning to cluster out the set of sources that are most likely to fall in the class of sources users is interested in. A static clustering algorithm, on the other hand, partitions data without any knowledge about the granularity of partitioning user's query may require.

Categorization approach is used in Web directory services such as Yahoo directory *dir.yahoo.com*. The static clustering or categorization scheme may not be able to handle the dynamic and ad-hoc nature of our problem setting. For example, such approaches may have either merged Music Records and Movies into same cluster, or might have partitioned Automobiles into imported cars, second-hand cars, and motor-cycles. Both of these can be undesirable in ad-hoc querying scenario, where partitions should be determined based on user requirements. Moreover, given the huge number of structured sources, if a clustering approach attempts to maintain the hierarchical partitioning of dataset with varying degrees of granularities, it would become very costly and totally unnecessary.

References

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

[2] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[3] BrightPlanet.com. The deep web: Surfacing hidden value. Accessible at `http://brightplanet.com/-deep_content/deepwebwhitepaper.pdf`, 2000.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of WWW7*, 1998.

[5] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.

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

[7] D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.

[8] L. Gravano, C.-C. K. Chang, H. García-Molina, and A. Paepcke. STARTS: Stanford protocol proposal for internet retrieval and search. Accessible at `http://www-db.stanford.edu/~gravano/-starts.html`, Aug. 1996.

[9] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *ACM SIGMOD*, 2003.

[10] H. He, W. Meng, C. T. Yu, and Z. Wu. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In *VLDB*, pages 357–368, 2003.

[11] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden-web: Hierarchical sampling and selection. In *VLDB*, 2002.

[12] P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: Categorizing hidden web databases. In *ACM SIGMOD*, 2001.

[13] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[14] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *VLDB*, 2001.

[15] J. D. Ullman. Information integration using logical views. Delphi, Greece, Jan. 1997.

[16] W. Wu, C. T. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *ACM SIGMOD*, pages 95–106, 2004.

[17] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *ACM SIGMOD*, 2004.