Discovering Attribute Locality across the Deep Web: an Ordering-Based Approach

Chengkai Li Kevin Chen-Chuan Chang

Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
cli@uiuc.edu, kcchang@cs.uiuc.edu

Abstract

The large number of structured database sources on the Web presents pressing need for information integration at a large scale. How can we enable systematic access to this "deep Web"? We observe that, while autonomous sources are seemingly independent, their query schemas often reveal certain correlations, such that sources in the same structured domain (e.g., books, cars) tend to share a "locality" of query attributes. This paper thus develops the notion of attribute localities, which is key for many schema-based integration tasks– such as source clustering and query mediation. Such attribute localities, while very useful, are computational expensive to discover. However, our observation further indicates that the localities are often self-revealing, when attributes are linearly ordered in a certain way, reflecting their connectivities. We thus further propose a novel ordering-based approach, which discovers localities by progressive construction, guided by attribute connectivities. Our experimental study shows that our notion of localities naturally capture the inherent structured domains, and our approach effectively discovers such localities, over hundreds of real Web sources.

1 Introduction

The Web has been rapidly "deepened" by the massive *searchable* Web databases: While the *surface Web* has linked billions of static HTML pages, a far more significant amount of information is hidden in the *deep Web*, behind the query forms of *searchable* databases. One example of deep Web sources is *Amazon.com*. These Web databases are often also referred to as the *hidden* or *invisible* Web. The July 2000 survey of [1] claims that there were 500 billion hidden pages in 10^5 online sources. Such information cannot be accessed directly through static URL links; they are only available as responses to dynamic queries submitted through the query interface of a database.

A wealth of work has been done in general information integration systems [2, 3], 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.* [4]).

We conduct an extensive survey [5] on deep Web to observe characteristics of the sources and study the implications of these characteristics on exploring and integrating Web databases. Although there are more structured databases on Web as we surveyed, relatively less work is done in this area as compared with text databases. The same challenges of large scale, which are equally important and difficult (if not more), exist for structured databases. We believe that the large number of structured sources on the Web presents pressing need for information integration at a large scale.

This paper focuses on *structured* databases on the deep Web, which provide structured information by accepting queries over the attributes on their query interfaces, or *schemas* (*e.g.*, author and title for *Amazon.com*). Thus, our focus essentially distinguishes *text* sources, which normally provide keyword-based search interfaces for text documents. Since data must be retrieved with queries, any attempts for integration must essentially interact with source schemas, which therefore are naturally the central notions for deep Web sources.

On the deep Web, we observe a distinguishing characteristic that offers a fresh view for schema-based large scale integration: while autonomous sources are seemingly independent, their query schemas often reveal certain correlations, such that sources in the same structured domain (e.g., books, cars) tend to share a "locality" of query attributes. Further, these localities naturally capture the inherent structured *domains* of sources.

This paper thus develops the notion of attribute localities, which is key for many schema-based integration tasks– such as source clustering, query expansion and source selection. For example, the natural correspondence between attribute localities and source domains allows us to cluster sources. Given a set of sources, we can discover the localities among all the attributes, therefore the domain of a source can be reconstructed by finding the localities of the attributes that are contained in this source. We postpone case studies on applications of attribute localities to Section 6.

The attribute localities, while very useful, are computational expensive to discover, because the number of possible localities configurations for a set of attributes is even larger than the number of possible partitions of these attributes. Therefore a brute force approach is not suitable for discovering the localities.

However, our observation further indicates that the localities are often self-revealing, when attributes are linearly ordered in a certain way. Inspired by this observation, we propose a novel ordering-based approach, which discovers localities by progressively constructing such an attribute order. Indeed, the characteristics revealed by the attribute order we observed can tell us how to construct it. We find that such an ordering reflects the connectivities between attributes. To be more specific, *first*, attributes belong to a locality gather together, highly connected to each other and weakly connected to attributes in other localities. *Second*, the attributes within a locality are roughly ordered according to their degree of connectivities to other attributes in the same locality. *Finally*, for an attribute at the border of two localities, it has few connectivities to the attributes in previous localities, but has high connectivities to the attributes in the succeeding locality.

Based on these observations, we develop an algorithm *COLD* (Connectivity-guided Ordering-based Locality Discovery) to discover attribute localities. It starts with an initial locality containing an initial attribute, constructs attribute order by greedily appending the attribute that has the highest connectivities to current locality, and detects the transition points between consecutive localities.

We also find that straightforward direct attribute connectivity is not a good measure. Considering the high importance of connectivity measure to robustness of the algorithm, we develop a new indirect connectivity measure, *context link*, that takes into consideration the contextual co-occurrences of attributes.

We performed experiments on about 500 real sources in 8 domains. The result shows that our notion of localities naturally capture the inherent structured domains, and our approach effectively discovers such localities.

To highlight, we summarize the main contributions of this paper as follows:

- We develop the **notion of attribute localities** based on an interesting phenomenon observed on the deep Web sources. Such attribute localities is key for many schema-based integration tasks.
- We propose a novel **ordering-based localities discovery approach**, inspired by the inherent characteristics of such localities.
- We present **connectivity-guided ordering construction algorithm** *COLD*, which progressively construct an attribute order according to the connectivities between attributes, also inspired by the characteristics of localities.
- We develop **context link** as an indirect attribute connectivity measure, which is more robust than straightforward direct connectivity.
- We report **experiments based on dataset of real Web sources**. Our experimental result verifies the effectiveness and practicality of our approach. We also conduct **case studies** to demonstrate the applications of attribute localities in the real world.

The remainder of the paper is organized as follows. Section 2 motivates the notion of attribute localities based on observations on the deep Web sources. Section 3 introduces the connectivity-guided ordering approach for localities discovery, the detailed algorithm for which is presented in Section 4. We report the experiments and evaluation in Section 5. Section 6 demonstrates the case studies on applications of attribute localities. Section 7 discusses related works. Finally, we make concluding remarks in Section 8. Appendix A shows the detailed experiment results.

2 Attribute Localities

With the virtually unlimited amount of information, the deep Web is clearly an important frontier for data integration. This "wild" frontier of the deep Web is characterized by its unprecedented scale. As a *challenge*: Sources online are virtually unlimited. Thus, any attempts to make the deep Web accessible and useful must cope with the challenge of large-scale integration. Second, as an *opportunity*: However, while sources proliferate, in aggregate, their complexity tends to be "concerted," revealing some underlying "structure". In particular, we observe that there exists naturally formed "localities" of attributes, corresponding to different source domains. Such attribute localities is very useful for exploring and integrating deep Web sources. In this section, we present our observation of localities and briefly discuss its applications, then we formally define the notion of "locality" and propose the problem of locality discovery.

2.1 Observation: Locality Phenomenon

We manually collected our dataset of deep Web sources using Web directories(e.g., *InvisibleWe-*

domain	*sources*	*domain*	*sources*
Airfares	53	Hotels	38
Automobiles	102	Jobs	55
Books	69	Movies	78
CarRentals	24	MusicRecords	75

Figure 1: *Deep Web Repository*:494 sources,8 domains.

b.com, *BrightPlanet.com*, *WebFile.com*) and search engines (e.g. *Google.com*). In particular, we collect 494 sources in eight domains, which are `Airfares(AI)`, `Automobiles(AU)`, `Books(B)`, `CarRentals(C)`, `Hotels(H)`, `Jobs(J)`, `Movies(MO)`, and `MusicRecords(MU)`. Figure 1 summarizes our dataset. We have released this dataset, the *Deep Web Repository*, available online at `http://eagle.cs.uiuc.edu/metaquerier`.

From this dataset, we observe a distinguishing characteristic that offers a fresh view for schema-based large scale integration: while autonomous sources are seemingly independent, their query schemas often reveal certain correlations, such that an attribute tends to relate to certain others, and they together form a *locality* of query attributes (*e.g.*, author, title, ISBN and publisher).

Further, these localities naturally capture the inherent structure *domains* of sources, such that sources in the same domain tend to share a "locality" (*e.g.*, author, title, ISBN, and publisher for the `Books` domain, and make, model, year for the `Automobiles` domain).

Figure 2(a) plots how attributes (the y-axis) occur in sources (the x-axis), so that a dot at (x, y) indicates that the schema of source x contains attribute y. Note that sources are ordered according to their domains, and attributes according to their order of first-occurrence along these ordered sources. Observe that each densely-dotted triangle along the diagonal represents an attribute locality, which is also squarely aligned with the domain boundaries of sources on the x-axis. The attributes that have occurrences only within some triangle are domain specific attributes. The attributes that have occurrences outside of the triangles are across domain.

As we have observed, we thus hypothesize the existence of attribute localities across sources. In fact, we believe such existence is to be expected: For each structured domain, naturally there is a conceptual pool of common attributes, or *vocabulary*, that characterizes the structured database objects exported by sources in that domain. These sources draw attributes from this vocabulary to form schemas. Therefore at a large scale with many sources of the same domain, we see their attributes form localities. Moreover, the localities correspond to different domains are naturally distinguishable from each other because the conceptual attribute pools for the domains are distinguishable, otherwise the source query interfaces can not tell users what kind of databases objects they export, thus would be confusing and useless to the users.

2.2 Definition of Locality

We have observed attribute localities in the above section, thus in this section we formally develop the notion of locality, which is key for many schema-based integration tasks– such as source clustering, query expansion and source selection. First, the natural correspondence between attribute localities and source domains enables identifying the domain of a source according to the localities of its schema attributes, therefore indirectly solves the problem of source clustering. Second, the localities enables expanding users' query attributes with relevant attributes in the localities. Last, based on source clustering and query expansion, source selection for answering users' queries is enabled.

We postpone the detailed case studies on these applications to Section 6. Below we formally define attribute locality and illustrate the concepts with examples.

As the building block of deep Web sources, query attributes are shown as the labels of query form elements on Web pages. The schema of a source contains a set of attributes. From now on, we will use schema and source interchangeably because we do not concern other features of a source in this paper.

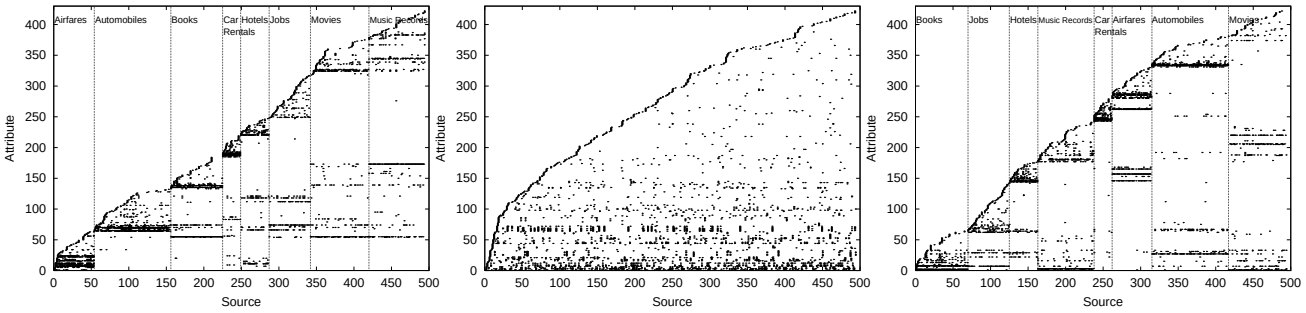
Definition 1 (Attribute-Source Space): An attribute-source space is a 3-tuple $(\mathcal{V}, \mathcal{S}, \mathcal{D})$: The vocabulary $\mathcal{V}$ is a set of attributes $\{a_1, ..., a_n\}$. $\mathcal{S}$ is a set of sources $\{s_1, ..., s_m\}$, where $s_i \subset \mathcal{V}$ for $1 \leq i \leq m$. Every attributes in $\mathcal{V}$ occur in some sources in $\mathcal{S}$, *i.e.*, $\mathcal{V} = \cup s_i$. $\mathcal{D}$ is a set of domains $\{d_1, ..., d_k\}$ that partitions $\mathcal{S}$, *i.e.*, $\mathcal{S} = \cup d_i$ and $d_i \cap d_j = \emptyset$. ∎

An example of attribute-source space is shown in Figure 3. There are three domains: `Automobiles`, `Books`, and `Movies`, each of which contains 10 sources and 10 attributes. There are altogether 25 attributes because there are attributes (*e.g.*, title) across multiple domains.

Definition 2 (Locality): A locality configuration on vocabulary $\mathcal{V}$, $l_\mathcal{V}$, is $\{l_1, ..., l_t\}$ where $l_i \subseteq \mathcal{V}$ for $1 \leq i \leq t$, and $\mathcal{V} = \cup l_i$. It is possible that $l_i \cap l_j \neq \emptyset$ for $l_i, l_j \in l_\mathcal{V}$.

$\mathcal{L}_\mathcal{V}$ is used to denote the set of all locality configurations on $\mathcal{V}$, *i.e.*, $\mathcal{L}_\mathcal{V} = \{l_\mathcal{V}\}$. ∎

Note that we consider domains as disjoint, which is a reasonable assumption based on the real world situation. However, we consider localities as overlapping, because there exist attributes that are common to multiple domains, and thus localities (*e.g.*, title in both `Books` and `Movies`).



(a) Locality: domain by domain. **(b)** No locality: shuffled across domains. **(c)** Locality: shuffled within domains.

Figure 2: Attribute locality phenomenon: attribute distributions over source domains.

Domain: Books
title, author, ISBN, publisher, price, subject, format, publication_Date, keyword, category

S_{b1}: (title, author, ISBN, publisher, price)
S_{b2}: (publisher subject, price, format, publication_Date)
S_{b3}: (title, author, price)
S_{b4}: (title, author, ISBN, keyword, format)
S_{b5}: (title, author)
S_{b6}: (title, ISBN, keyword, category)
S_{b7}: (title, author, ISBN, keyword)
S_{b8}: (title, author, ISBN, publisher, subject)
S_{b9}: (title, author, keyword, publisher, price)
S_{b10}: (title, keyword, category)

Domain: Automobiles
make, model, year, style, type, zip_code, price, color, state, area

S_{a1}: (make, model, year, style, type)
S_{a2}: (make, zip_code)
S_{a3}: (make, model, price, year, color)
S_{a4}: (make, model, price, zip_code, type)
S_{a5}: (make, model, price, year)
S_{a6}: (price, zip_code)
S_{a7}: (make, model, price, year, zip_code, state)
S_{a8}: (make, year, zip_code)
S_{a9}: (make, price, type, area)
S_{a10}: (make, model, price, year, type)

Domain: Movies
title, actor, director, price, genre, rating, format, type, cast/crew, studio

S_{m1}: (title, actor, director, price)
S_{m2}: (title, actor, director, genre, rating, price, format, type)
S_{m3}: (title, actor, director, genre)
S_{m4}: (actor, director, rating, format type)
S_{m5}: (title, format, cast/crew, studio)
S_{m6}: (actor, director, type, studio)
S_{m7}: (actor, director, genre, rating)
S_{m8}: (title, cast/crew)
S_{m9}: (title, genre, price, format, studio)
S_{m10}: (title, actor, director)

Figure 3: Example of attribute-source space

locality 1	locality 2	locality 3
title	title	make
author	actor	model
ISBN	director	year
publisher	price	style
price	genre	type
subject	rating	zip_code
format	format	price
publication_date	type	color
keyword	cast/crew	state
category	studio	area

Figure 4: Example of attribute localities

Definition 3 (Domain Attributes Projection): For an attribute-source space $(\mathcal{V}, \mathcal{S}, \mathcal{D})$, the domain attributes projection, $l_{\mathcal{D}}$, is $\{l_1, ..., l_k\}$, where $l_i = \cup_{s_j \in d_i} s_j$. ∎

Given the observed existence of attribute localities and its promising applications, our problem is to uncover such a hidden locality configuration, $l_{\mathcal{V}} \in \mathcal{L}_{\mathcal{V}}$ for the attribute-source space $(\mathcal{V}, \mathcal{S}, \mathcal{D})$, with $\mathcal{V}$ and $\mathcal{S}$ as given and $\mathcal{D}$ as hidden. $\mathcal{L}_{\mathcal{V}}$ is the set of all possible locality configurations, among which there are good and poor ones. The domain attributes projection $l_{\mathcal{D}}$ is an instance of locality configuration, *i.e.*, $l_{\mathcal{D}} \in \mathcal{L}_{\mathcal{V}}$. For example, domain attributes projection for the attribute-source space in Figure 3 is shown in Figure 4. Naturally it is a good locality configuration because it is generated according to the real domains of sources. However, without domain knowledge, it is non-trivial to discover such a locality configuration, as we discuss in the section below.

2.3 Difficulties in Locality Discovery

The locality phenomenon is self-revealing only when we order sources domain by domain, as we did in Figure 2(a). Without such knowledge of domains, it is non-trivial to discover locality. Figure 2(b) plots how attributes (the y-axis) occur in sources (the x-axis) in the same way as in Figure 2(a), but the sources on the x-axis are randomly from all domains instead of domain by domain. Obviously we can only see a single sparse triangle covering all the sources and all the attributes, instead of localities.

Consider attributes as nodes in an *attribute graph*, in which two attributes (nodes) are connected by an edge if they co-occur in some schemas. The edge is weighted by the number of co-occurrences of the two attributes. Imagine the extreme case that all domains of sources are isolated from each other without any shared attribute, and all the sources in a domain have the exact same schema. In such a case, each attribute locality is a "perfect" clique on the graph.

However, the concept of locality as "clique" would be *fuzzy* rather than definite because the real world is much more complex than the ideal case. Above we assume there exists "perfect" localities which have two properties: fully connected attributes within locality, and isolated localities. It is clear that there is even no single such "perfect" clique in the real world. Therefore it is nontrivial to find the fuzzy "cliques", *i.e.*, localities.

First, the attributes within a locality are not fully connected. To begin with, two attributes in a locality may not co-occur frequently or even do not co-occur. Moreover, two attributes with few co-occurrences are not necessarily weakly associated. Instead, they could be strongly associated if they both heavily co-occur with other common attributes in the locality. Final-

ly, the degree of association for each attribute pair is different.

Second, the attributes from different localities are related instead of isolated. As shown in Figure 2(a), there exists a lot of attributes that occur in multiple domains, which we name *linking* attributes. The existence of linking attributes reflects the natural semantic connections between different domains. For example, attribute 55 in Figure 2(a) is title, which occurs in multiple domains such as `Books`, `Movies`, and `MusicRecords`.

Furthermore, such linking attributes are abundant, frequent and they are the only constituents of many sources, as we observed from the dataset. First, there are a large amount of linking attributes. Among the 422 attributes in all sources, 17.5%(74/422) are linking attributes. Second, these linking attributes occur frequently. Intuitively, the more domains an attribute occurs in, the more frequent it is. Among the top 20 frequent attributes, 18 attributes are linking attributes. Third, the linking attributes constitute a large amount of sources. 21.9% (108/494) sources contain only linking attributes.

3 Locality Discovery: Connectivity-Guided Ordering

Attribute localities are difficult to uncover as we discussed in Section 2.3. However, we further observe that the localities are often self-revealing, when attributes are linearly ordered in a certain way. Indeed, the characteristics revealed by the attribute order we observed can tell us how to construct it. We find that such an order reflects the connectivities between attributes. Inspired by this observation, we propose a novel connectivity-guided ordering-based approach, which discovers localities by progressively constructing such an attribute order.

3.1 Observation: Attribute Ordering

From Figure 2(a), we observe two interesting attribute *ordering* effects, which are *macro ordering* and *micro ordering* respectively.

Macro ordering: Attributes belong to a locality gather together and localities appear on the y-axis in the order corresponding to the order of domains on the x-axis. Therefore we observe that each densely-dotted triangle along the diagonal is squarely aligned with the domain boundaries of sources on the x-axis.

Micro ordering: the attributes within a locality are roughly ordered according to their numbers of co-occurrences with other attributes in the same locality. Therefore we see the thick bottom and sparse top of triangles, because the vertical alignment of two attributes represents their co-occurrence.

Moreover, the order of observing domains and the order of observing sources within a domain does not have much impact on these two attribute ordering effects. Figure 2(c) plots how attributes (the y-axis) occur in sources (the x-axis) in the same way as in Figure 2(a), but the domains are in a different order from Figure 2(a) and sources within each domain are observed with a random different order. Interestingly, we still observe the *ordering* effects.

3.2 Locality Discovery: Connectivity-Guided Greedy Ordering

The *ordering* effects shown by the localities in Figure 2(a) motivate us to find localities along the order of attributes. Once we find a "magic" attribute order as in Figure 2(a), the localities themselves would be self-revealed. What is left to do is just detecting the transition points at the border of consecutive domains.

Suppose there are altogether n attributes ($|\mathcal{V}| = n$), then the number of possible ordering schemes is $n!$. This prohibiting large number makes it infeasible a naive approach of evaluating all possible orderings.

Indeed, the characteristics revealed by the "magic" order in Figure 2(a) can tell us how to construct it. We find that the attribute order reflects the connectivities between attributes. To be more specific, *first*, attributes gathered in the same locality are highly connected to each other and weakly connected to attributes in other localities. Therefore we observe the dense triangles and relatively sparse region outside of the triangles. *Second*, the micro ordering effect tells us that attributes within a locality are roughly ordered according to their number of co-occurrences with other attributes in the same locality. The number of co-occurrences reflect connectivities. *Finally*, for an attribute at the border of two localities, it has few connectivities to the attributes in previous localities, but has high connectivities to the attributes in the succeeding locality.

According to the above observations, we propose to greedily construct an attribute order guided by attribute connectivities for discovering localities. Specifically, assume we have a function $\mathcal{C}$ to evaluate the connectivities between an attribute and a locality, starting from one locality $\mathcal{L}$ containing one attribute a (the most frequent attribute, as in our algorithm), the following three rules can be used to construct a sequence of attributes:

Rule 1: The next attribute to be added into the sequence, a_{next}, is the one that has the highest connectivities to attributes in current locality $\mathcal{L}$, *i.e.*, $a_{next} = argmax_a \mathcal{C}(a, \mathcal{L})$.

Rule 2: If there is no attribute that has connectivities to current locality $\mathcal{L}$, *i.e.*, $\mathcal{C}(a_{next}, \mathcal{L}) = 0$, then a new attribute will start a new locality, which becomes the current locality.

Rule 3: Suppose a is the last attribute and a' is the second to last attribute in the sequence. If both $\mathcal{C}(a_{next}, \mathcal{L})$ and $\mathcal{C}(a', \mathcal{L} \setminus \{a', a\})$ is higher than $\mathcal{C}(a, \mathcal{L} \setminus \{a\})$ (which means a is highly connected to a_{next} but less connected to other attributes in $\mathcal{L}$), then a and a_{next} start a new locality, which becomes the current locality.

We introduce the details of the algorithm in Section 4, including the function $\mathcal{C}$ mentioned above. Below we introduce how to compute the connectivities between two attributes, which is the foundation of $\mathcal{C}$.

3.3 Context Link: Serving as Connectivity

Mutual information is a measure of the amount of information one random variable contains about another [6], therefore could be used to capture the correlation between two attributes. Suppose $p(x = 1)$ is the probability that attribute x appear in a schema and $p(x = 1, y = 1)$ is the probability that attribute x and y appear together in a schema. The mutual information of x and y is $I(x;y) = \sum_{x=0}^{1}\sum_{y=0}^{1} p(x,y) log\frac{p(x,y)}{p(x)p(y)}$.

However the correlation expressed by mutual information does not capture the connectivity that we desire. The reason is that mutual information considers presence the same important as absence in schema data, while connectivity only considers co-occurrences as important. Mutual information combines four situations: both x and y are presented, none of them is presented, x but y is presented, and y but x is presented. For example, it considers x and y highly correlated if x always occur in a schema when y does not occur.

Therefore we define connectivity of two attributes as following. Instead of combining the four situations, it only measures whether the probability of x and y co-occurring in a schema is significant. The extra 1 is for the purpose of making the result value be over 0.

Definition 4 (Connectivity): $c(x, y|\mathcal{V}, \mathcal{S}) = p(x = 1, y = 1) log(1 + \frac{p(x=1,y=1)}{p(x=1)p(y=1)})$. ∎

This definition is based on the direct connectivities of two attributes. However, attributes may be indirectly connected.

Example 1: Suppose there are two domains of schemas, `Books` and `Movies`, as shown in Figure 5. The first domain contains 5 attributes title, author, ISBN, subject, and format. The second domain contains 6 attributes title, actor, director, genre, rating, and format. Note that although subject and format do not co-occur at all, they both co-occur with other attributes. Therefore although their direct connectivity is zero, they should have high indirect connectivity. On the other hand, although format and rating co-occur once, they do not both co-occur with many common attributes.

Domain: Books title, author, ISBN, subject, format	Domain: Movies title, actor, director, genre, rating, format
(title, author, ISBN, subject) (title, author, ISBN, format) (title, author, subject) (title, author, format) (author, ISBN, subject) (author, ISBN, format)	(title, actor, director) (title, genre, rating, format) (title, actor, director, genre) (title, actor, director, rating) (actor, director, genre)

Figure 5: Example of indirect connectivity

Therefore their indirect connectivity is low. Moreover, if we take into consideration that title occurs in most sources, then it should endow less connectivity between format and rating. ∎

As a better similarity measure for two transactions than direct similarity such as *Jaccard coefficient*, "link" is defined in [7] as the number of common neighbors of two transactions. Inspired by this idea, we extend the concept of connectivity of attributes to *context link*.

Definition 5 (Context Link): For a vocabulary $\mathcal{V}$ and a set of sources $\mathcal{S}$, the context link between two attributes $x, y \in \mathcal{V}$, $link(x, y|\mathcal{V}, \mathcal{S})$, is defined as the sum of their direct link and indirect links, *i.e.*, $link(x, y|\mathcal{V}, \mathcal{S}) = c(x, y|\mathcal{V}, \mathcal{S}) + \sum_{i=1}^{+\infty} w_i \cdot link_i(x, y|\mathcal{V}, \mathcal{S})$, where $link_i(x, y|\mathcal{V}, \mathcal{S}) = \sum_{a_1,...,a_i \in \mathcal{V}, a_1,...,a_i \neq x,y} min(c(x, a_1|\mathcal{V}, \mathcal{S}), c(a_1, a_2|\mathcal{V}, \mathcal{S}), ..., c(a_i, y|\mathcal{V}, \mathcal{S}))$. $link_i(x, y|\mathcal{V}, \mathcal{S})$ is *context i-link* of x and y, which is the indirect link between x and y through other i attributes, *i.e.*, x and a_1 co-occur, a_2 and a_3 co-occur, ..., a_i and y co-occur. All the i-links are weighted by some w_i. ∎

The computation of full context link is exponentially complex, therefore we adopt only context 1-link with weight 1, which experimentally is shown to be a good selection. Thus, $link(x, y|\mathcal{V}, \mathcal{S}) = c(x, y|\mathcal{V}, \mathcal{S}) + \sum_{a \in \mathcal{V}, a \neq x,y} min(c(x, a|\mathcal{V}, \mathcal{S}), c(a, y|\mathcal{V}, \mathcal{S}))$.

Example 2: For the attribute-source space in Example1, $c(\text{format}, \text{subject}) = 0$ and $c(\text{format}, \text{rating}) = 0.113$, but $link(\text{format}, \text{subject}) = 0.738$ and $link(\text{format}, \text{rating}) = 0.425$. ∎

4 Algorithm *COLD*: Connectivity-guided Ordering-based Locality Discovery

In this section, we introduce the Algorithm *COLD* (Connectivity-guided Ordering-based Locality Discovery) for discovering attribute localities, which starts with an initial locality containing an initial attribute, constructs attribute order by greedily appending the attribute that has the highest connectivities to current locality, and detects the transition points between consecutive localities. The algorithm is shown in Figure 6. It consists of five steps: *filtering*, *ordering*, *reallocating*, *multiple allocating*, and *putting back filtered attributes*. Below we explain the details of each step one by one.

Algorithm Locality Discovery
Input:
- $\mathcal{V}$: input attributes

Output:
- $\mathcal{L}$: localities

Procedure:
01. (1) **Filtering Rare Attributes**:
02. $\mathcal{V}' \leftarrow \{v \mid v \in \mathcal{V}, frequency(v) > 1\}$
03. (2) **Ordering Attributes**:
04. $\mathsf{a}_{start} \leftarrow \text{argmax}_{\mathsf{a} \in \mathcal{V}'} frequency(\mathsf{a})$
05. $\mathcal{L} \leftarrow Order(\mathcal{V}', \mathsf{a}_{start})$
06. (3) **Reallocating**:
07. $\mathcal{L}' \leftarrow \{l | \exists l' \in \mathcal{L}, link(l, l') > localitiness(l)\}$
08. $\mathcal{L} \leftarrow Reallocate(\mathcal{L} \setminus \mathcal{L}', \mathcal{L}')$ //TLH
09. Reallocating based on SRH (Section 4.3)
10. $\mathcal{L}' \leftarrow \{l | l \in \mathcal{L}, l.size() = 1\}$
11. $\mathcal{L} \leftarrow Reallocate(\mathcal{L} \setminus \mathcal{L}', \mathcal{L}')$ //SLH
12. (4) **Multiple Allocating**:
13. for $l \in \mathcal{L}$
14. $l \leftarrow l \cup \{a | a \in \mathcal{V}, a \notin l, link(a, l) > localitiness(l)\}$
15. (5) **Putting Back Filtered Attributes**:
16. $\mathcal{L} \leftarrow Reallocate(\mathcal{L}, \{\mathcal{V} \setminus \mathcal{V}'\})$

Figure 6: Algorithm: Locality Discovery

4.1 Step 1: Filtering

This step filters those attributes that occur only in one source. Empirically we find that these rare attributes weaken the robustness of the following steps of the algorithm. Therefore we do not let these attributes participate in the rest steps. Instead, we put back these attributes in the last step. Intuitively since such a rare attribute only occur in one schema, its localities can easily be determined according to other attributes in that schema.

Example 3: From now on, we use the small example of attribute-source space in Figure 3 to illustrate the algorithm. The results for the attribute-source space of the dataset we collect are reported in Appendix. The example in Figure 3 contains 3 domains, each of which contains 10 sources and 10 attributes. In the filtering step, publication_data, state, style, color, and area are filtered because they only occur once. ■

4.2 Step 2: Ordering

The details of the function *Order* in Figure 6 is shown in Figure 7. It starts with the attribute of highest frequency, as the initial locality. As motivated in Section 3, the attributes ordering is greedily constructed by iteratively adding attribute that has the highest connectivities to current locality. The connectivity between two attributes is captured by the *context link* defined in Section 3.3, based on which the connectivities between an attribute and a locality is defined as *attribute-locality link*, shown below.

Algorithm $Order(\mathcal{V}, a_{start})$
Input:
- $\mathcal{V}$: input attributes.
- a_{start}: starting attribute.

Output:
- $\mathcal{L}$: localities.

Procedure:
01. $\mathcal{L} \leftarrow \{\}$; $\mathcal{CL} \leftarrow \{a_{start}\}$; $\mathcal{V} \leftarrow \mathcal{V} \setminus \{a_{start}\}$
02. while $\mathcal{V}$ is not empty
03. $a_{max} \leftarrow argmax_{a \in \mathcal{V}}\ link(a, \mathcal{CL})$
04. if $link(a_{max}, \mathcal{CL}) = 0$ then
05. $\mathcal{L}.insert(\mathcal{CL})$
06. $a_{start} \leftarrow argmax_{a \in \mathcal{V}} frequency(a)$
07. $\mathcal{CL} \leftarrow \{a_{start}\}$; $\mathcal{V} \leftarrow \mathcal{V} \setminus \{a_{start}\}$
08. else
09. if $\mathcal{CL}.size() > 2$ then
10. $a \leftarrow \mathcal{CL}[\mathcal{CL}.size()]$; $a' \leftarrow \mathcal{CL}[\mathcal{CL}.size() - 1]$
11. if $link(a', \mathcal{CL} \setminus \{a', a\}) > link(a, \mathcal{CL} \setminus \{a\})$
12. and $link(a_{max}, \mathcal{CL}) > link(a, \mathcal{CL} \setminus \{a\})$
13. then
14. $\mathcal{L}.insert(\mathcal{CL} \setminus \{a\})$
15. $\mathcal{CL} \leftarrow \{a\}$
16. $\mathcal{CL}.insert(a_{max})$; $\mathcal{V} \leftarrow \mathcal{V} \setminus \{a_{max}\}$
17. $\mathcal{L}.insert(\mathcal{CL})$
18. return $\mathcal{L}$

Figure 7: Algorithm *Order*

Definition 6 (Attribute-Locality Link): Given a locality $l = \{a_1, ..., a_k\}$ and an attribute $a \notin l$, the *attribute-locality link* between a and l, $link(a, l) = \frac{1}{k} \sum_{i=1}^{k} link(a, a_i)$. ■

The function *Order* has two facilitating data structures. $\mathcal{L}$ is used to record all the discovered localities. $\mathcal{CL}$ is the current locality. Under two situations, $\mathcal{CL}$ is inserted into $\mathcal{L}$ and a new $\mathcal{CL}$ is formed. First is when there is no attribute that has connectivity to $\mathcal{CL}$, therefore no attribute can be added into it. After $\mathcal{CL}$ is inserted into $\mathcal{L}$, the most frequent attribute among the rest attributes is selected as the initial attribute of a new $\mathcal{CL}$, just the same as how the ordering procedure is started. Second is when a transition attribute is detected (introduced below). In this case, the transition attribute is treated as the starting attribute of the new $\mathcal{CL}$.

4.2.1 Transition Attributes

Transition attributes are those attributes on the border of two consecutive localities, defined below according to the third characteristics of the "magic" order as we described in Section 3.2.

Definition 7 (Transition Attribute): Given a current locality $\mathcal{CL} = \{a_1, a_2, ..., a_k\}$, where attributes

$a_1, a_2, ..., a_k$ are in the order by which they are added into $\mathcal{CL}$. a_{max} is the attribute that has the highest connectivity to $\mathcal{CL}$, *i.e.*, $\mathsf{a}_{max} = argmax_{\mathsf{a} \in \mathcal{V}'}$ *link*$(\mathsf{a}, \mathcal{CL})$, where $\mathcal{V}'$ is set of attributes that have not been included into the attribute ordering yet. before. a_k is a transition attribute if the following two conditions are satisfied:
(a) *link*$(a_{k-1}, \mathcal{CL} \setminus \{a_{k-1}, a_k\}) >$ *link*$(a_k, \mathcal{CL} \setminus \{a_k\})$
(b) *link*$(a_{max}, \mathcal{CL}) >$ *link*$(a_k, \mathcal{CL} \setminus \{a_k\})$
Note that $\mathcal{CL}$ contains at least 3 attributes if a_k is a transition attribute. ∎

After the transition attribute a_k is detected, it is removed from the current locality $\mathcal{CL}$, which is inserted into $\mathcal{L}$. Then the new current locality $\mathcal{CL}$ is set to be $\{a_k, a_{max}\}$.

Attribute a_{max} is well connected to the transition attribute a_k, as justified below. *link*$(a_k, \mathcal{CL} \setminus \{a_k\}) >$ *link*$(a_{max}, \mathcal{CL} \setminus \{a_k\})$ because a_k is the attribute that has highest attribute-locality link to $\{a_1, ..., a_{k-1}\}$. Therefore *link*$(a_{max}, \mathcal{CL}) >$ *link*$(a_{max}, \mathcal{CL} \setminus \{a_k\})$ according to (b) in Definition 7. *link*$(a_{max}, \mathcal{CL}) = \frac{\textit{link}(a_k, a_{max}) + (k-1) \cdot \textit{link}(a_{max}, \mathcal{CL} \setminus \{a_k\})}{k}$, therefore we have *link*$(a_k, a_{max}) >$ *link*$(a_{max}, \mathcal{CL} \setminus \{a_k\})$. This means a_{max} has high connectivity to a_k but not to the rest of $\mathcal{CL}$. Meanwhile, according to (a) in Definition 7, a_k does not have high connectivity to $\mathcal{CL}$, at least not as high as its previous attribute a_{k-1}.

Based on the analysis above, it is reasonable to start a new locality with a_k and a_{max}. Even if the decision is incorrectly made (a_k should be in $\mathcal{CL}$ or in both), it can be remedied later by multiple allocation in Step 4.

Example 4: In the example of Figure 3, the most frequent attribute is title, therefore the algorithm starts with it. The localities found by ordering is shown in Figure 8(a). Right after each attribute, the domains in which the attribute has occurrences are listed (A for `Automobiles`, B for `Books`, and M for `Movies`). Note that 4 localities are discovered instead of 3. We name these localities as l_1, l_2, l_3 and l_4 for ease of representations. Attributes in 1_1 all occur in `Books`, attributes in l_2 all occur in `Movies`, and attributes in l_3 all occur in `Automobiles` except subject. Locality l_4 is not a good one, which contains one attribute from `Books` and one attribute from `Movies`. No linking attributes are discovered. We show in later steps how these problems can be removed. We also visualize the discovered localities by plotting the attribute-locality link of each attributes to its current locality, as shown in Figure 9. There are 3 transition attributes, which are actor, make and category. ∎

4.3 Step3: Reallocating

The localities formed in Step 2 may not be good and attributes may not be in appropriate localities. We solve these problems by detecting inappropriate localities and attribute assignments according to the following three heuristics, and reallocating the attributes.

	locality 1	locality 2	locality 3	locality 4
(a)	title (B/M) author (B) ISBN (B) keyword (B) publisher (B) price (B/A/M) format (B/M)	actor (M) director (M) genre (M) rating (M) type (A/M) studio (M)	make (A) model (A) year (A) zip_code (A) subject (B)	category (B) cast/crew (M)
(b)	title (B/M) author (B) ISBN (B) keyword (B) publisher (B) price (B/A/M) format (B/M) category (B)	actor (M) director (M) genre (M) rating (M) type (A/M) studio (M) cast/crew (M)	make (A) model (A) year (A) zip_code (A) subject (B)	
(c)	title (B/M) author (B) ISBN (B) keyword (B) publisher (B) price (B/A/M) format (B/M) category (B) subject (B)	actor (M) director (M) genre (M) rating (M) type (A/M) studio (M) cast/crew (M)	make (A) model (A) year (A) zip_code (A)	
(d)	title (B/M) author (B) ISBN (B) keyword (B) publisher (B) price (B/A/M) format (B/M) category (B) subject (B)	actor (M) director (M) genre (M) rating (M) type (A/M) studio (M) cast/crew (M) title (B/M) format (B/M)	make (A) model (A) year (A) zip_code (A)	
(e)	title (B/M) author (B) ISBN (B) keyword (B) publisher (B) price (B/A/M) format (B/M) category (B) subject (B) publication_date (B)	actor (M) director (M) genre (M) rating (M) type (A/M) studio (M) cast/crew (M) title (B/M) format (B/M)	make (A) model (A) year (A) zip_code (A) state (A) style (A) area (A) color (A)	

(a) Ordering; **(b)**Reallocating: TLH; **(c)**Reallocating: SRH; **(d)**Multiple Allocating; **(e)**Putting Back.

Figure 8: Results of *COLD* for the example in Figure 3

The three heuristics are all based on a *Reallocate* function, shown in Figure 10, which assign an attribute to the locality that has the highest attribute-locality link with the attribute.

Tight Locality Heuristic (TLH): This heuristic is for identifying those illy formed localities. It requires that attributes within a locality should be well connected to each other, corresponding to the first characteristics of the "magic" order as we described in Section 3.2. All the attributes in locality l is reallocated if there exists another locality l' such that the *inter-locality link* between l and l' is higher than the localitiness of l, *i.e.*, *link*$(l, l') >$ *localitiness*(l). The localitiness and inter-locality link are defined as below.

The *localitiness* of a locality quantifies the degree of links between attributes within a locality.

Definition 8 (Localitiness): Given a locality l, its *localitiness* is *localitiness*$(l) = \frac{2}{|l| \cdot (|l|-1)} \sum_{x,y \in l, x \neq y}$ *link*(x, y). ∎

The *inter-locality link* quantifies the degree of links between attributes from two localities.

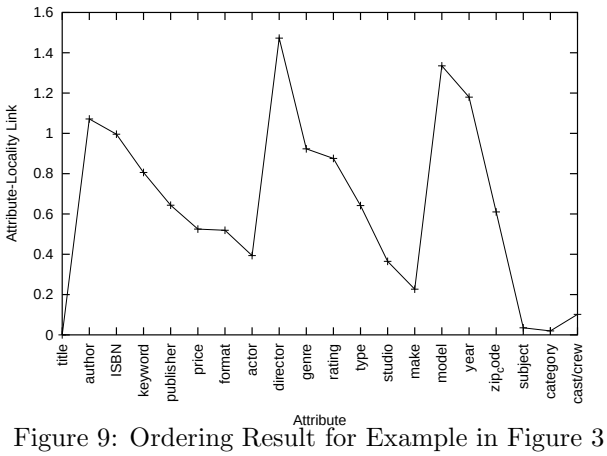


Figure 9: Ordering Result for Example in Figure 3

Algorithm *Reallocate*$(\mathcal{L}, \mathcal{L}')$
Input:
- $\mathcal{L}$: localities to be remained
- $\mathcal{L}$': localities to be reallocated

Output:
- $\mathcal{L}$: new localities

Procedure:
01. for $l' \in \mathcal{L}'$
02. for $\mathsf{a} \in l'$
03. $l_{max} \leftarrow argmax_{l \in \mathcal{L}} link(\mathsf{a}, l)$
04. $l_{max} \leftarrow l_{max} \cup \{\mathsf{a}\}$
05. return $\mathcal{L}$

Figure 10: Algorithm *Reallocate*

Definition 9 (Inter-Locality Link): Given two localities l and l', the *inter-locality link* between l and l' is $link(l, l') = \frac{1}{|l| \cdot |l'|} \sum_{x \in l, y \in l'} link(x, y)$. ■

Example 5: For example, in Figure 8(a), l_4 (category and cast/crew) is obviously not good. According to tight locality heuristic, this locality is reallocated because there exists locality, *e.g.*, l_1, such that $link(l_1, l_4) > localitiness(l_4)$. category and cast/crew are moved to l_1 and l_2 respectively using *Reallocate*$(\{l_1, l_2, l_3\}, \{l_4\})$. The resulting localities are shown in Figure 8(b). ■

Source Representing Heuristic (SRH): This heuristic is for identifying those inappropriately allocated attributes. It requires that for an attribute in a locality, some source that contains the attribute must corresponds to the same locality. It first corresponds sources to localities by the following method. A source $s = \{a_1, ..., a_k\}$, treated as a virtual locality, is determined to correspond to locality $l = argmax_{l_i \in \mathcal{L}} link(s, l_i)$, *i.e.*, the locality that has the largest inter-locality link with s. If there exists an attribute a in locality l, such that there is no source containing a correspond to l, then reallocate attribute a.

Example 6: For example, in Figure 8(b), subject is in l_3 which contains mostly attributes from the `Automobiles` domain. According to source representing heuristic, among the sources that are assigned to l_3 by the above source-locality correspondence method, there is no source that contains subject. Therefore subject is reallocated to l_1. The resulting localities are in Figure 8(c). ■

Small Locality Heuristic (SLH): This is a simple heuristic, which requires that the size of each locality should at least be 2. Those localities with only one attribute is reallocated.

4.4 Step4: Multiple Allocating

As we find in Section 2.3, there are linking attributes which occur in multiple domains, therefore should be in multiple localities. We also find that these linking attributes are large-amount, frequent, and constituting many sources, thus should not be disregarded. Therefore partitional locality configurations such as Figure 8(a)(b)(c) should be avoided. The algorithm allocate attributes to multiple localities based on the following heuristic.

Multiple Allocating Heuristic (MAH): A attribute a is allocated to a locality l if the attribute-locality link is higher than localitiness of l, *i.e.*, $link(a, l) > localitiness(l)$.

Example 7: The result of allocating attributes to multiple localities is shown in Figure 8(d). To be specific, title and format are allocated to domain `Movies`, in addition to `Books`. ■

4.5 Step5: Putting Back Rare Attributes

Those rare attributes filtered in Step 1 are put back to suitable localities using the reallocating function *Reallocate*in Figure 10.

Example 8: The result of putting back rare attributes is shown in Figure 8(e). We can see that all the rare attributes (publication_data, state, style, color, and area) are put back to proper localities. ■

5 Experiments

We conduct experiments of Algorithm *COLD* on the deep web sources repository we collected at `http://-eagle.cs.uiuc.edu/metaquerier`. There are 494 sources in eight domains, as summarized in Figure 1, and 422 attributes altogether. We manually extracted attributes from the schemas of these sources. We also did some straightforward preprocessing to merge attributes of slight textual variations (*e.g.*, authors and author). We focus on discovering attribute locality and consider such attribute extraction and preprocessing as independent tasks. In particular, attribute extraction (*e.g.*, extracting title from "please input title") can

locality	main domain	others
1	B:50	MO:5 MU:4 MO/MU:2
2	MO/MU:19 MO:11 MU:10	0
3	MU:44	0
4	MO:41	AU:1
5	H:34	MU:1
6	J:77	C:6 H:4
7	AI:60	AU:2
8	C:29	0
9	AU:55	0

Figure 11: Discovered localities on *Deep Web Repository*

be automated with noun-phrase extraction tools, such as LinkIT [8].

Due to the space limitation, the localities discovered on this attribute-source space is shown in Figure 12(in Appendix). We evaluate the results in Figure 11. Our algorithm finds 9 localities, which is very close to the number of domains, 8, without any parameters given by human experts. For each discovered locality, most of its attributes are in one domain, and there are few attributes from other domains allocated to this locality. For example, all the 55 attributes in locality 9 are from `Automobiles` (either domain-specific attributes or linking attributes), and there is no attribute from other domain. We also find that all the attributes that only occur in one domain are also allocated to the corresponding localities. Therefore all the attributes counted under the column "others" in Figure 11 are at the same time also allocated to the appropriate localities. For example, the two attributes from `Automobiles` that are allocated to locality 7 are also multiply allocated to locality 9.

Localities 2, 3, and 4 worth special discussions. Note that our algorithm does not require any parameter such as number of localities, therefore 9 localities are discovered although there are 8 domains. Locality 2 corresponds to attributes that occur in `Movies` or `MusicRecords`, locality 3 corresponds to `MusicRecords`, and locality 4 corresponds to `Movies`. Indeed, such a configuration is not necessarily bad. By checking our dataset, we find that there are a lot of schemas accepting queries for both movies and music records, although some may emphasize movies and others emphasize music records. However, we manually assigned these sources to `Movies` or `MusicRecords`, without realizing beforehand that there should be a third domain for such kind of sources. The discovering of locality 2 indicates the localities found very naturally correspond to real world domains, even subtle one.

6 Case Study: Applications

In this section, we introduce three case studies demonstrating the applications of attribute localities in integrating deep Web sources.

Source Clustering: Figure 2 indicates that there is a natural mapping between localities and domains. The discovered localities enable us to cluster deep web sources based on their schemas. For example, for the localities $\mathcal{L} = \{l_1, ..., l_n\}$, construct domains $\mathcal{D} = \{d_1, ..., d_n\}$, where d_i is corresponding to l_i. A source $s = \{a_1, ..., a_k\}$, treated as a virtual locality, is assigned to the domain corresponding to the locality that has the largest inter-locality link with the s, *i.e.*, $domain(s) = d_i$, where $i = argmax_{1 \le i \le n} link(s, l_i)$. (Note that this domain assignment uses the same approach as source representing heuristic in Section 4.3.) The domains of the sources in Figure 3 are fully recovered by this method.

Existing Web clustering approaches rely on not only page contents, but also link structure, anchor text and other features of HTML, therefore are very expensive compared with using source schemas alone for clustering. Moreover, these approaches do not deal with query interfaces expressly, therefore the contents of pages that contain the query interfaces incur noisy information, instead of revealing the domains of the structured databases exported by the interfaces. [9] classifies text databases with keyword query interfaces by sending probing queries to the sources to sample the contents of sources, which is also very expensive.

Query Expansion: Query expansion technique in IR and Web search expands users' original query with new terms and re-weights the terms in the expanded query [10]. Its motivation is the phenomenon that users often need to spend large amounts of time reformulating their queries to accomplish effective retrieval because of the lack of detailed knowledge of the retrieval environment. Therefore query expansion treats the original query as an initial attempt and expands it by examining the retrieved documents from the original query.

Analogously users may feel helpless facing the large amount of query attributes of the deep Web sources because she does not know which attribute to query. Even if she has some attribute in mind, she does not know which other attributes should she use to further confine her queries because it is tedious to browse all the source schemas to figure out the connections between attributes.

Given the discovered attribute localities, it is easy to bring up the locality of an attribute. The size of the returned attributes can be controlled by ranking the links between the attribute and other attributes in the locality. We can view this as giving the user an "advanced query form" based on the input of just one single attribute. This facility not only eases the user's effort to specify query attributes, but also can tell her useful attributes that she may even do not know before.

Source Selection: Based on source clustering and query expansion, source selection is enabled. Suppose

a user wants to query a set of attributes $\{a_1, ..., a_k\}$, the related localities is $\{l | \exists a_i \in l, l \in \mathcal{L}, 1 \leq i \leq k\}$. Therefore multiple advanced query forms will be provided to users.

For each locality, the sources that could answer the expanded queries can be automatically queried after the user fill the "advanced query form". As we discussed before, sources are clustered into domains corresponding to localities. The query results of these sources can be unioned with the help of wrapping technique [11].

There may be linking attributes across the localities. In that case, these linking attributes are presented to the user, who further identifies those linking attributes that she thinks are useful. Consider the case that there are 2 localities. For such a linking attribute a, from the two corresponding domains d_1 and d_2, two sources $s_1 \in d_1$ and $s_2 \in d_2$ that both contain a are considered to be joinable. Therefore the query results of s_1 and s_2 can be joined.

Example 9: For the attribute-source space in Figure 3, the localities discovered are in Figure 8(e). The domains are fully recovered using the method introduced in source clustering (therefore we will just use the hidden domains in Figure 3 to refer to the domains constructed by clustering.) Suppose a user provides three attributes title, author, and director. The localities l_1 and l_2 are identified to be relevant to user's query, which correspond to two domains `Books` and `Movies` respectively. Linking attributes title and format are presented to the user, who specifies that title should be treated as join attribute. The user further specifies that author and director should be queried with values she provides. In addition, hinted by the returned localities, she specifies that subject and genre should also be queried. Therefore query $(s_{b8} \bowtie s_{m2}) \cup (s_{b8} \bowtie s_{m3})$ is automatically determined as the final query. After she fills the values for those attributes to query, pairs of book (from s_{b8}) and movie (from s_{m2} or s_{m3}) that have the same title specified by her and satisfy other attribute values (author and subject for s_{b8}, director and genre for s_{m2} and s_{m3}) are returned to her. ∎

7 Related Works

Database sources on the Web has gained much attention from the community recently. Ipeirotis et al. [9, 12] classify and summarize contents of text databases by sending probing queries to the sources. Raghavan et al. [13] have built a crawler for the deep web. Davulcu et al. [14] propose a 3-layer architecture for querying Web sources with navigational query interfaces. [11] discusses the wrapper generation techniques, targeting at extraction of semantic information out of HTML pages populated by backend databases.

Much work is focusing on schema matching, which takes two schemas as input and produces a mapping between elements of the two schemas that correspond semantically to each other. The work on this topic is surveyed in [15].

We can abstract the schema data studied in this paper as market basket data (or transaction data) if we view an attribute as a product item, and a schema of attributes as a transaction of items. Association rules [16] are used to identify frequently associated products in market basket data. However, association rules does not capture the concept of localities. As discussed in [17], association rules can only represent very fine-grained knowledge, thus too many valid association rules would be found if applied on schemas, and each rule contains a small number of attributes. Conceptually, association rules only captures strong perfect cliques in the attribute graph mentioned in Section 2.3 because all attributes in a frequent attribute set have to co-occur over the "support" [16].

Clustering is a technique for grouping data into clusters so that data points within a cluster have high similarity and data points across different clusters have high dissimilarity. Surveys of different clustering algorithms can be found in [18, 19]. Although bearing similarities to clusters, the concept of *locality* distinguishes from cluster a lot. First of all, traditional clustering algorithms are not designed for clustering transaction data. Second, they cluster data points instead of the attributes of the points.

ROCK [7] is a clustering algorithm for transaction and categorical attributes. It defines a novel similarity measure for transactions, *link*, which uses the common neighbors between two transactions as similarity instead of using direct similarity measure such as Jaccard coefficient. Based on link, ROCK employs an agglomerative hierarchical clustering algorithm to cluster transactions and categorical data.

The concept of link inspires us to develop context link between attributes in this paper. However, ROCK [7] requires many carefully selected parameters such as number of clusters, threshold for defining neighbors based on which link is defined, and a important function defining the criterion function, which measures the goodness of clusters. Moreover, the hierarchical clustering methods generate a tree presentation of the results, which has large size. Without specifying the parameters such as number of clusters and merging threshold, the tree presentation cannot be turned into partitional result, and thus is difficult to digest. However, our algorithm generate localities which naturally correspond to domains.

[17] develops a method for clustering items in transaction data based on association rule hypergraph model. It first find frequent itemsets, based on which a weighted hypergraph is constructed. Hypergraph partitioning [20] is performed on the association rule hypergraph to find clusters. To achieve good clustering results, this approach requires carefully selected pa-

rameters. Only frequent attributes will participate in clustering. This approach also assumes no overlapping items across clusters, which contrasts with the significant linking attributes in schema data.

8 Conclusions

In this paper, we develop a notion of attribute localities for schemas of deep Web sources. While autonomous sources are seemingly independent, their query schemas often reveal certain correlations, such that sources in the same structured domain tend to share a "locality" of query attributes. Such localities, while useful for many integration tasks, are difficult to discover. However, we find that the localities are self-revealing, when attributes are linearly ordered in a way reflecting their connectivities. Inspired by these observations, we propose a novel ordering-based approach for discovering localities, which greedily and progressively construct an attribute order according to the connectivities between attributes. The experimental results of our algorithm *COLD* on real Web sources verifies the effectiveness and practicality of our approach. Three case studies on applications of attribute localities shows that attribute localities is promising for enabling systematic access to the structured deep Web sources.

References

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

[2] J. D. Ullman. Information integration using logical views. In *Proceedings of the 6th International Conference on Database Theory*, Delphi, Greece, Jan. 1997. Springer, Berlin.

[3] 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.

[4] 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.

[5] K. C.-C. Chang, B. He, C. Li, and Z. Zhang. Strucutred databases on the web: Observations and implications. Technical Report UIUCDCS-R-2003-2321, Department of Computer Science, UIUC, Feb. 2003.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[7] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, pages 512–521. IEEE Computer Society, 1999.

[8] D. K. Evans, J. Klavans, and N. Wacholder. Document processing with linkit. In *Proc. of the RIAO Conference, Paris, France*, 2000.

[9] P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: Categorizing hidden web databases. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, Santa Barbara, Ca., May 2001.

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

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

[12] P. G. Ipeirotis and L. Gravano. Distributed search over the hidden-web: Hierarchical database sampling and selection. In *Proc. 28th Int. Conf. Very Large Data Bases, VLDB*, 2002.

[13] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of VLDB 2001*. Morgan Kaufmann, 2001.

[14] H. Davulcu, J. Freire, M. Kifer, and I. V. Ramakrishnan. A layered architecture for querying dynamic web content. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM Press, 1999.

[15] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[16] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[17] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering based on association rule hypergraphs. In *SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 9–13, 1997.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[19] J. Han and M. Kamer. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[20] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: application in vlsi domain. In *Proceedings of the 34th annual conference on Design automation conference*, pages 526–529. ACM Press, 1997.

Appendix

locality 1

Title (AI AU B J MO MU)
ISBN (B MO)
Author (B MO)
Keyword (AU B J MO MU)
Subject (B MO)
Format (B MO MU)
Publisher (B MU)
Price (AU B H MO MU)
Category (AI AU B J MO MU)
Binding (B)
Section (B)
Content (B MU)
Topic (B)
Person (MO)
Book Code (B)
Theater (MO)
Last Name (B)
First Name (B)
Description (B MO MU)
Item Number (B MU)
Publication Year (B MU)
Series (AU B)
First Edition (B)
Signed (B)
Dust Jacket (B)
Publication Date (B)
Print Status (B)
Reader Age (B)
Shipping Destination (B)
Language (B MO)
Date Added (MO MU)
Music Category (MU)
People (MO)
Director (MO MU)
Alibris ID (B)
Added Within (B)
Soundex (B)
Award (B)
Review Source (B)
Vendor (B)
Audience (B)
Imprint (B)
CBD Stock Number (B)
User Level (B)
Availability (B)
Catelog Number (B)
Item Description (B)
Book Class (B)
Annotation (B)
Lexile (B)
Interest Level (B)
Reading Level (B)
Dewey Decimal (B)
Copyright (B)
Reading Program (B)
Review (B)
Movie (MO)
Code Number (MO)
KeyWord (MU)
Singer (MU)
Player (MU)

locality 2

Director (MO MU)
Actor (MO MU)
Genre (MO MU)
Rating (AU MO)
Movie Title (MO MU)
Artist (B MO MU)
Synopsis (MO)
Group (B MO MU)
Media (B MO MU)
Song Track (MO MU)
Inventory (MO MU)
Condition (MO MU)
Band (MU)
Software (J MU)
Cast/Crew (MO MU)
Decade (MO)
Screenwriter (MO)
Region Code (MO)
Category (AI AU B J MO MU)
Title (AI AU B J MO MU)
Keyword (AU B J MO MU)
Format (B MO MU)
Catalog Number (MO MU)
Studio (MO)
Label (MO MU)
Album (MO MU)
Producer (MO MU)
Album Title (MO)
Song Title (MO)
Movie Soundtrack (MO)
DVD Title (MO)
Character (MO)
Similar Artist (MU)
Video (MU)
Photo (MU)
Station (MU)
Hardware (MU)
News (MU)
Rington (MU)
Tour Date (MU)

locality 3

Catalog Number (MO MU)
Label (MO MU)
Composer (MO MU)
Performer (B MU)
Orchestra (MU)
Conductor (MU)
Media Type (B MU)
Musician (MU)
Song (MU)
Album (MO MU)
CD Title (MU)
Ensemble (MU)
Work Title (MU)
Instrument (MU)
Soundtrack (MU)
Release Year (MO MU)
Style (AU MU)
Record Label (MU)
Sub Category (MU)
Music Type (MU)
Title (AI AU B J MO MU)
Format (B MO MU)
Artist (B MO MU)
Recording Name (MU)
Venue (MU)
Recording Quality (MU)
Guest (MU)
Video Title (MU)
DVD (MU)
Track (MU)
Album Info (MU)
Raga (MU)
Speed (MU)
Date Added (MU)
Track Artist (MU)
Track Title (MU)
CD Release Year (MU)
Barcode (MU)
Guest Artist (MU)
Catalog Code (MU)
Release Month (MU)
Composition (MU)
Piece (MU)
Soloist (MU)

locality 4

Producer (MO MU)
Studio (MO)
Writer (MO MU)
Department (B MO)
Part Number (AU MO)
Release Date (B MO)
Starring (MO)
DVD Release Year (MO)
Theatrical Release Year (MO)
Special Feature (MO)
Closed Caption (MO)
Category (AI AU B J MO MU)
Title (AI AU B J MO MU)
Price (AU B H MO MU)
Rating (AU MO)
Format (B MO MU)
Actor (MO MU)
Director (MO MU)
Genre (MO MU)
Date (AU)
Language Audio Track (MO)
Picture Format (MO)
Subtitle (MO)
Audio Type (MO)
Certificate (MO)
Wide Screen (MO)
Keywords (MO)
Regional Coding (MO)
Video Format (MO)
Audio Encoding (MO)
Production Year (MO)
Subtitle Language (MO)
Audio Language (MO)
Version (MO)
Tomatometer (MO)
Plot (MO)
Legend (MO)
Editor (MO)
Cinematographer (MO)
Sound (MO)
Running Time (MO)
Extra Info (MO)

locality 5

Check Out Date (H)
Check In Date (H)
Number of Rooms (H)
Airport Code (C H)
Property Type (H)
Adults per Room (H)
Hotel Chain (H)
Hotel Name (H)
Amenities (H)
Smoking (H)
Bed Type (H)
Number of Beds (H)
Number of Guests (H)
Rate (H)
Number of Nights (H)
Lodging Type (H)
Arrival Date (H)
Room Type (H)
Facilities (H)
Radius (AU H J)
Currency (AI B C H)
Number of Adults (AI H)
State (AU B C H J MU)
Postal/Zip Code (AU C H J)
City (AU B C H J MO)
Number of Cribs (H)
Star Rating (H)
Arriving Day (H)
Hotel Brand (H)
Address (H)
Details (H)
Activities (H)
Hotel Type (H)
Local Region (H)
Date Uploaded (MU)

locality 6

State (AU B C H J MU)
City (AU B C H J MO)
Country(AU B C H J MO MU)
Profession (J)
End Date (H J)
Start Date (C H J)
Industry Sector (J)
Job Level (J)
Sector (J)
Travel (J)
Date Listed (J)
Industry Category (J)
Position Type (J)
Company Name (J)
Location (AU B C H J)
Job Type (J)
Salary (J)
Industry (J)
Education (J)
Job Title (J)
Company Type (J)
Experience Level (J)
Experience (J)
Date Posted (J)
Discipline (J)
Degree (J)
Job Category (J)
Job Skill (J)
Date Range (J)
Employment Type (J)
Company (J)
Postal/Zip Code (AU C H J)
Keyword (AU B J MO MU)
Check In Date (H)
Check Out Date (H)
Hotel Name (H)
Number of Rooms (H)
Car Group (C)
Start Time (C)
Finish Date (C)
Finish Time (C)
Delivery to Address (C)
Puck Up Location (C)
Job ID (J)
Work Type (J)
Job Detail (J)
Job Location (J)
Career Type (J)
Provice (J)
Corporate Partner (J)
Commute (J)
Role Title (J)
Tax Term (J)
Field Experience (J)
Length (J)
Specialty (J)
Position Category (J)
Management (J)
Entry Level (J)
Sate (J)
Job Area (J)
Employment Level (J)
Job Duration (J)
Employer Name (J)
General Vicinity (J)
Skill (J)
Business Travel (J)
Schedule (J)
Company Size (J)
Job Posted By (J)
Visa Sponsorship (J)
Field (J)
Duration (J)
Organization Type (J)
Position Description (J)
Job Freshness (J)
Time Period (J)
Job Sector (J)
Travel Level (J)
Relocation Cost Paid (J)
Employer (J)
Skill Title (J)
Function (J)
Job Description (J)
Mode (J)
Job Date (J)
Role (J)

locality 7

Number of Adults (AI H)
Number of Children (AI H)
Departure Date (AI H)
Return Date (AI C)
Departure Time (AI)
Return Time (AI C)
Destination City (AI)
Area Code (AU)
Brand (AU)
Leaving From (AI)
Going To (AI)
One Way (AI)
Round Trip (AI)
Number of Infants (AI)
Number of Seniors (AI H)
Class (AI AU)
Airline (AI C)
Number of Passengers (AI)
Departure City (AI)
Cabin (AI)
Destination (AI H)
Arrival City (AI)
NonStop (AI)
Service Class (AI)
Number of Connections (AI)
Number of Stops (AI)
Number of Travelers (AI)
Travel Class (AI)
Fare Type (AI)
Return City (AI C)
Cabin Class (AI)
Origin (AI)
Number of Tickets (AI)
Vacation Destination (AI)
Cruise Destination (AI)
Cruise Ship (AI)
Type of Class (AI)
Number of Babes (AI)
Frequent Flyer Number (AI)
Multi Cities (AI)
Leaving On (AI)
Returning On (AI)
eCertificate (AI)
Pricing Option (AI)
Avoid Change Penalties (AI)
No Advance Purchase-
Restrictions (AI)
Direct Flights Only (AI)
Preferred Airlines (AI)
Departing From (AI)
Departing Date (AI)
Booking Class (AI)
Fare (AI)
Class of Service (AI)
Reference Code (AI)
Certificate Number (AI)
Maximum Bid Amount (AI)
Auction Number (AI)
Package Type (AI)
Auction Ending Before (AI)
Promotion Code (AI)
Number of Youth (AI)
Multiple Destinations (AI)

locality 8

Flight Number (C)
Pick Up Date (C)
Pick Up Time (C)
Drop Off Date (C)
Drop Off Time (C)
Pick Up Location (C)
Drop Off Location (C)
Pick Up City (C)
Car Type (AU C)
Drop Off City (C)
Air Conditioning (C)
Transmission (AU C)
Car Rental Company (C)
Car Class (C)
Drop Off Country (C)
Pick Up Country (C)
Country of Residence (C)
Car Category (C)
Driver's Age (C)
Preferred Agencies (C)
Return Location (C)
Street (C H)
Vehicle Class (C)
Rental Location (C)
Pick Up Point (C)
Car Transmission (C)
Vicinity (C)
Rental Car Company (C)
Travelers (C)

locality 9

Postal/Zip Code (AU C H J)
Year (AU MO)
Make (AU)
Model (AU)
Mileages (AU)
New (AU)
Used (AU B MO MU)
Distance (AU J)
Body Style (AU)
Color (AU)
Options (AU)
Number of Cylinders (AU)
Number of Doors (AU)
Fuel Type (AU)
Pre-Owned (AU)
Vehicle Type (AU)
Engine Size (AU)
Body Type (AU)
Vehicle (AU)
Payment Method (AU)
County (AU)
Dealer (AU)
Stock Number (AU MO)
Type (AU B MO MU)
Manufacturer (AU MO MU)
Region (AU J MU)
Postcode (AU)
Area (AU H J)
E-mail Address (AU)
Size (AU H)
Price (AU B H MO MU)
Transmission (AU C)
Awards (AU)
Item Location (AU)
Internet Price (AU)
Body (AU)
Lease Period (AU)
Monthly Payment (AU)
Drive Type (AU)
Number of Seats (AU)
Fuel Efficiency (AU)
Features (AU)
Odometer Reading (AU)
Seller Type (AU)
Listing Type (AU)
Sale Type (AU)
Engine (AU)
Number of Sears (AU)
Trade or Private (AU)
Registration Year (AU)
Drive Wheels (AU)
Purchase Time (AU)
Type of Vehicle (AU)
Model Description (AU)
Classification (AU)

Figure 12: Results of the Algorithm *COLD* for sources in our Deep Web Repository