Discovering Complex Matchings across Web Query Interfaces: A Correlation Mining Approach*

Bin He, Kevin Chen-Chuan Chang, Jiawei Han
Computer Science Department
University of Illinois at Urbana-Champaign
binhe@uiuc.edu, {kcchang, hanj}@cs.uiuc.edu

ABSTRACT

To enable information integration, schema matching is a critical step for discovering semantic correspondences of attributes across heterogeneous sources. While complex matchings are common, because of their far more complex search space, most existing techniques focus on simple 1:1 matchings. To tackle this challenge, this paper takes a conceptually novel approach by viewing schema matching as *correlation mining*, for our task of matching Web query interfaces to integrate the myriad databases on the Internet. On this "deep Web," query interfaces generally form *complex matchings* between attribute groups (e.g., {author} corresponds to {first name, last name} in the Books domain). We observe that the co-occurrences patterns across query interfaces often reveal such complex semantic relationships: *grouping attributes* (e.g., {first name, last name}) tend to be co-present in query interfaces and thus positively correlated. In contrast, *synonym attributes* are negatively correlated because they rarely co-occur. This insight enables us to discover complex matchings by a correlation mining approach. In particular, we develop the DCM framework, which consists of *data preparation*, *dual mining* of positive and negative correlations, and finally *matching selection*. Unlike previous correlation mining algorithms, which mainly focus on finding strong positive correlations, our algorithm cares both positive and negative correlations, especially the subtlety of negative correlations, due to its special importance in schema matching. This leads to the introduction of a new correlation measure, H-measure, distinct from those proposed in previous work. We evaluate our approach extensively and the results show good accuracy for discovering complex matchings.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases; H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Measurement

*This material is based upon work partially supported by NSF Grants IIS-0133199 and IIS-0313260. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.

KDD'04, August 22–25, 2004, Seattle, Washington, USA.

Keywords

data integration, deep Web, schema matching, correlation mining, correlation measure

1. INTRODUCTION

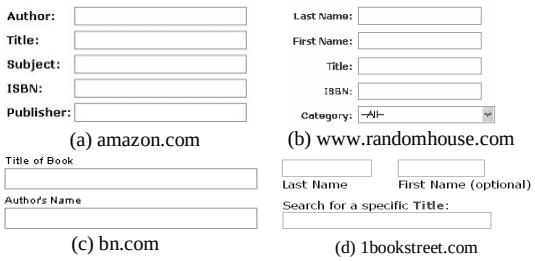
In recent years, we have witnessed the rapid growth of databases on the Web, or the so-called "deep Web." A July 2000 survey [3] estimated that 96,000 "search cites" and 550 billion content pages in this deep Web. Our recent study [6] in December 2002 estimated between 127,000 to 330,000 deep Web sources. With the virtually unlimited amount of information sources, the deep Web is clearly an important frontier for data integration.

Schema matching is fundamental for supporting query mediation across deep Web sources. On the deep Web, numerous online databases provide dynamic *query*-based data access through their *query interfaces*, instead of static URL links. Each query interface accepts queries over its *query schemas* (e.g., author, title, subject, ... for *amazon.com*). Schema matching (i.e., discovering semantic correspondences of attributes) across Web interfaces is essential for mediating queries across deep Web sources.

In particular, matching Web interfaces in the same domain (e.g., Books, Airfares), the focus of this paper, is an important problem with broad applications. In particular, we often need to search over alternative sources in the same domain such as purchasing a book (or flight ticket) across many online book (or airline) sources. Given a set of Web interfaces in the same domain, this paper solves the problem of discovering matchings among those interfaces. We notice that our input, a set of Web pages with interfaces in the same domain, can be either manually [7] or automatically [13, 12] collected and classified.

On the "deep Web," query schemas generally form *complex matchings* between attribute groups. In contrast to simple 1:1 matching, complex matching matches a set of m attributes to another set of n attributes, which is thus also called $m{:}n$ *matching*. We observe that, in query interfaces, complex matchings do exist and are actually quite frequent. For instance, in Books domain, author is a synonym of the grouping of last name and first name, i.e., {author} = {first name, last name}; in Airfares domain, {passengers} = {adults, seniors, children, infants}. Hence, discovering complex matchings is critical to integrate the deep Web.

Although 1:1 matching has got great attention [18, 9, 15, 10], $m{:}n$ matching has not been extensively studied, mainly due to the much more complex search space of exploring all possible combinations of attributes (as Section 7 will discuss). To tackle this challenge, we investigate the *co-occurrence* patterns of attributes across sources, to match schemas *holistically*. Unlike most schema matching work which matches *two* schemas at a time, we match *all* the schemas at the same time. This holistic matching provides the co-occurrence information of attributes across schemas and thus



(a) amazon.com (b) www.randomhouse.com

(c) bn.com (d) 1bookstreet.com

Figure 1: Examples of "Fragment" Web Interfaces.

enables efficient mining-based solutions. For instance, we may observe that last name and first name often co-occur in schemas, while they together rarely co-occur with author, as Figure 1 illustrates. More generally, we observe that *grouping attributes* (i.e., attributes in one group of a matching e.g., {last name, first name}) tend to be co-present and thus positively correlated across sources. In contrast, *synonym attributes* (i.e., attribute groups in a matching) are negatively correlated because they rarely co-occur in schemas.

These dual observations motivate us to develop a correlation mining abstraction of the schema matching problem. Specifically, given Web pages containing query interfaces, this paper develops a streamlined DCM framework for mining complex matchings, consisting of automatic *data preparation* and *correlation mining*, as Figure 2 shows. Since the query schemas in Web interfaces are not readily minable in HTML format, as preprocessing, the data preparation step prepares "schema transactions" for mining (Section 5). Then the correlation mining step, the main focus of this paper, discovers complex matchings with dual mining of positive and negative correlations (Section 3). We name the whole matching process as DCM, since the core of the algorithm is the dual correlation mining part.

Unlike previous correlation mining algorithms, which mainly focus on finding strong positive correlations, our algorithm cares both positive and negative correlations. Hence, we need to develop measures for both positive correlations and negative ones. Our schema matching task is particularly interested in negative correlations, since on one hand, they reflect the synonym relationships among attributes, on the other hand, they have not been extensively explored and applied before.

To ensure the quality of the mining result (i.e., the complex matchings), the chosen measures should satisfy some quality requirements, based on our observation of query schemas (Section 4). In particular, from the extremely non-uniform distribution of schema attributes, we identify that: 1) Both the positive and negative correlation measures should be robust for the *sparseness problem* (i.e., the sparseness of schema data may "exaggerate" the effect of co-absence), which has also been noticed as the "null invariance" property by recent correlation mining work [20, 16, 14]. 2) The negative correlation measure should be robust for the *rare attribute problem* (i.e., the rare attributes may not be convincing to judge their negative correlations). Since none of the existing measures [20, 4] is robust for both the sparseness problem and the rare attribute problem, we develop a new measure, H-measure, robust against both problems in measuring negative correlations.

To evaluate the matching performance and H-measure, we test the DCM framework on the datasets in the UIUC Web integration repository [7]. First, we test DCM on the TEL-8 dataset, which contains raw Web pages over 447 deep Web sources in 8 popular domains, and the result shows good *target accuracy*. Second, we compare the DCM framework with the MGS framework [10], which also matches Web interfaces with the same insight of exploring a holistic approach, on its BAMM dataset. The result shows that DCM is empirically close to MGS in discovering simple matchings and further DCM can find complex matchings, which is not supported by MGS. Third, we compare H-measure with other measures on the TEL-8 dataset and the result shows H-measure outperforms the others in most cases.

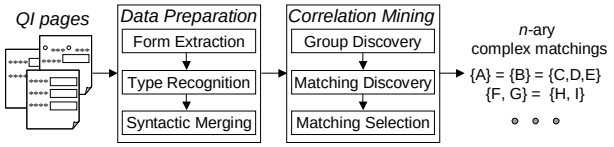


Figure 2: From matching to mining: the DCM framework.

There are several applications of our work: First, while pursuing holistic matching, our result can naturally address the pairwise matching problem. For instance, given the matching {author} = {last name, first name} found by our approach, we can match {author} in some schema S_A to {last name, first name} in another schema S_B. Second, our work is a critical step to construct a global Web interface for each domain. Specifically, among the synonyms in a matching, we can pick the most popular one as the representative in the global interface and use that matching to build the mappings from the global interface to local ones.

In our development, we also observed several interesting issues. Can we mine interesting patterns over cross-domain Web interfaces? How to systematically decide the threshold values for mining? How can our approach benefit from exploring other information on the Web? We discuss these open issues in Section 8.

In summary, the contributions of this paper are:

- We build a conceptually novel connection between the schema matching problem and the correlation mining approach. On one hand, we consider schema matching as a new *application* of correlation mining; on the other hand, we propose correlation mining as a new *approach* for schema matching.
- We develop correlation measures that are robust for not only positive correlations, but also negative correlations. In particular, we identify the problems of existing measures on evaluating negative correlations, due to its special importance in schema matching, and further introduce a new correlation measure, H-measure, distinct from those proposed in previous work.

The rest of the paper is organized as follows: Section 2 presents our motivating observations of integrating the deep Web. Section 3 develops the mining and selection algorithms. Section 4 proposes a new correlation measure, H-measure. Section 5 presents the data preparation step. Section 6 reports our experiments. Section 7 reviews related work and and Section 8 discusses several further opportunities and open issues, and then concludes this paper.

2. MOTIVATION: FROM MATCHING TO MINING

As Section 1 briefly introduced, our key insight is on connecting matching to mining, which this section further motivates with a concrete example. Consider a typical scenario: suppose user Amy, who wants to book two flight tickets from city A to city B, one for her and the other for her 5-year old child. To get the best deal, she needs to query on various airfare sources by filling the Web query interfaces. For instance, in *united.com*, she fills the query interface with from as city A, to as city B and passengers as 2. For the same query in *flyairnorth.com*, she fills with depart as city A, destination as city B, adults as 1, seniors as 0, children as 1 and infants as 0.

This scenario reveals some critical characteristics of the Web interfaces in the same domain. First, some attributes may *group* together to form a "larger" concept. For instance, the grouping of adults, seniors, children and infants denotes the number of passengers. We consider such attributes that can be grouped as *group-*

ing attributes or having *grouping relationship*, denoted by putting them within braces (e.g., {adults, seniors, children, infants}).

Second, different sources may use different attributes for the same concept. For instance, from and depart denote the city to leave from, and to and destination the city to go to. We consider such semantically equivalent attributes (or attribute groups) as *synonym attributes* or having *synonym relationship*, denoted by "=" (e.g., {from} = {depart}, {to} = {destination}).

Grouping attributes and synonym attributes together form *complex matchings*. In complex matching, a set of m attributes is matched to another set of n attributes, which is thus also called m:n *matching*, (in contrast to the simple 1:1 matching). For instance, {adults, seniors, children, infants} = {passengers} is a 4:1 matching in the above scenario.

To tackle the complex matching problem, we exploit the co-occurrence patterns to match schemas *holistically* and thus pursue a mining approach. Unlike most schema matching work which matches two schemas at a time, we match all the schemas at the same time. This holistic view provides the co-occurrence information of attributes across many schemas, which reveals the semantics of complex matchings. (Such co-occurrence information cannot be observed when schemas are matched only in pairs.) For instance, we may observe that adults, seniors, children and infants often co-occur with each other in schemas, while they together do not co-occur with passengers. This insight enables us to discover complex matchings with a correlation mining approach. In particular, in our application, we need to handle not only positive correlations, a traditional focus, but also negative ones, which have rarely been extensively explored or applied.

By matching many schemas together, this holistic matching naturally discovers a more general type of complex matching – a matching may span across more than two attribute groups. Still consider the Amy scenario, if she tries a third airline source, *priceline.com*, she needs to fill the interface with departure city as city A, arrival city as city B, number of tickets as 2. We thus have the matching {adults, seniors, children, infants} = {passengers} = {number of tickets}, which is a 4:1:1 matching. Similarly, we have two 1:1:1 matchings {from} = {departure city} = {depart} and {to} = {arrival city} = {destination}. We name this type of matching n*-ary complex matching*, which can be viewed as an aggregation of several binary m:n matchings.

In particular, we develop a new approach, the DCM framework, to mine n-ary complex matchings. Figure 2 illustrates this mining process: 1) As preprocessing, data preparation (Section 5) prepares "schema transactions" for mining by extracting and cleaning the *attribute entities* in Web interfaces. 2) As the main focus of this paper, the correlation mining step (Section 3) discovers n-ary complex matchings by first finding potential attribute groups using positive correlations and then potential complex matchings using negative correlations. Last, matching selection chooses the most confident and consistent matchings from the mining result. 3) Also, since pursuing a mining approach, we need to choose appropriate correlation measures. We discuss this topic in Section 4.

3. COMPLEX MATCHING AS CORRELATION MINING

We view a schema as a *transaction*, a conventional abstraction in association and correlation mining. In data mining, a transaction is a set of items; correspondingly, in schema matching, we consider a schema as a set of *attribute entities*. An attribute entity contains attribute name, type and domain (i.e., instance values). Before mining, the data preparation step (Section 5) finds syntactically similar entities among schemas. After that, each attribute entity is assigned a unique *attribute identifier*. While the mining is over the attribute entities, for simplicity of illustration, we use the attribute name of each entity, after cleaning, as the attribute identifier. For instance, the schema in Figure 1(c) is thus, as a transaction of two attribute entities, written as {title, author}.

Formally, we consider the schema matching problem as: *Given the input as a set of schemas $\mathcal{S}_\mathcal{I} = \{S_1, ..., S_u\}$ in the same domain, where each schema S_i is a transaction of attribute identifiers, find all the matchings $\mathcal{M} = \{M_1, ..., M_v\}$. Each M_j is an n-ary complex matching $G_{j_1} = G_{j_2} = ... = G_{j_w}$, where each G_{j_k} is an attribute group and $G_{j_k} \subseteq \bigcup_{t=1}^{u} S_i$.* Semantically, each M_j should represent the synonym relationship of attribute groups G_{j_1},..., G_{j_w} and each G_{j_k} should represent the grouping relationship of attributes in G_{j_k}.

Motivated by our observations on the schema data (Section 2), we develop a correlation mining algorithm, with respect to the above abstraction (Figure 2). *First, group discovery*: We mine *positively correlated attributes* to form potential attribute groups. A potential group may not be eventually useful for matching; only the ones having synonym relationship (i.e., negative correlation) with other groups can survive. For instance, if all sources use last name, first name, and not author, then the potential group {last name, first name} is not useful because there is no matching (to author) needed. *Second, matching discovery*: Given the potential groups (including singleton ones), we mine *negatively correlated attribute groups* to form potential n-ary complex matchings. A potential matching may not be considered as correct due to the existence of conflicts among matchings. *Third, matching selection*: To solve the conflicts, we develop a selection strategy to select the most confident and consistent matchings from the mining result. Section 3.1 discusses the group and matching discovery and Section 3.2 the matching selection.

After group discovery, we need to add the discovered groups into the input schemas $\mathcal{S}_\mathcal{I}$ to mine negative correlations among groups. (A single attribute is viewed as a group with only one attribute.) Specifically, an attribute group is added into a schema if that schema contains any attribute in the group. For instance, if we discover that last name and first name have grouping relationship, we consider {last name, first name} as an attribute group, denoted by G_{lf} for simplicity, and add it to any schema containing either last name or first name, or both. The intuition is that although a schema may not contain the entire group, it still partially covers the concept that the group denotes and thus should be counted in matching discovery for that concept. Note that we do not remove singleton groups {last name} and {first name} when adding G_{lf}, because G_{lf} is only a potential group and may not survive in matching discovery.

3.1 Complex Matching Discovery

While group discovery works on individual attributes and matching discovery on attribute groups, they can share the same mining process. We use the term – *items* – to represent both attributes and groups in the following discussion of mining algorithm.

Correlation mining, at the heart, requires a measure to gauge correlation of a set of n items; our observation indicates pairwise correlations among these n items. Specifically, for n groups forming synonyms, any two groups should be negatively correlated, since they both are synonyms by themselves (e.g., in the matching {destination} = {to} = {arrival city}, negative correlations exist between any two groups). We have similar observation on the attributes with grouping relationships. Motivated by such observations, we design the measure as:

$$C_{min}(\{A_1, ..., A_n\}, m) = \min m(A_i, A_j), \forall i \neq j, \qquad (1)$$

where m is some correlation measure for two items (e.g., the measures surveyed in [20]). That is, we define C_{min} as the minimal

Algorithm: N-ARYSCHEMAMATCHING:
Input: InputSchemas $\mathcal{S}_\mathcal{I} = \{S_1, ..., S_u\}$,
Measures m_p,m_n, Thresholds T_p, T_n
Output: Potential n-ary complex matchings
begin:
1 /* group discovery */
2 $\mathcal{G} \leftarrow$ APRIORICORRMINING$(\mathcal{S}_\mathcal{I}, m_p, T_p)$
3 /* adding groups into $\mathcal{S}_\mathcal{I}$ */
4 **for** each $S_i \in \mathcal{S}_\mathcal{I}$
5 **for** each $G_k \in \mathcal{G}$
6 **if** $S_i \cap G_k \neq \emptyset$ **then** $S_i \leftarrow S_i \cup \{G_k\}$
7 /* matching discovery */
8 $\mathcal{M} \leftarrow$ APRIORICORRMINING$(\mathcal{S}_\mathcal{I}, m_n, T_n)$
9 **return** $\mathcal{M}$
end

Algorithm: APRIORICORRMINING:
Input: InputSchemas $\mathcal{S}_\mathcal{I} = \{S_1, ..., S_u\}$,
Measures m, Thresholds T
Output: Correlated items
begin:
1 $X \leftarrow \emptyset$
2 $\mathcal{V} \leftarrow \bigcup_{t=1}^{u} S_i, S_i \in \mathcal{S}_\mathcal{I}$
3 **for** all $A_p, A_q \in \mathcal{V}, p \neq q$
4 **if** $m(A_p, A_q) \geq T$ **then** $X \leftarrow X \cup \{\{A_p, A_q\}\}$
5 $l \leftarrow 2$
6 /* X_l: correlated items with length = l */
7 $X_l \leftarrow X$
8 **while** $X_l \neq \emptyset$
9 construct X_{l+1} from X_l using apriori feature
10 $X \leftarrow X \cup X_{l+1}$
11 $X_l \leftarrow X_{l+1}$
12 **return** X
end

Figure 3: Algorithm N-ARYSCHEMAMATCHING.

value of the pairwise evaluation, thus requiring all pairs to meet this minimal "strength."

C_{min} has several advantages: First, it satisfies the "apriori" feature and thus enables the design of an efficient algorithm. In correlation mining, the measure for qualification purpose should have a desirable "apriori" property (i.e., downward closure), to develop an efficient algorithm. (In contrast, a measure for ranking purpose should not have this "apriori" feature, as Section 3.2 will discuss.) C_{min} satisfies the "apriori" feature since given any item set $\mathcal{A}$ and its subset $\mathcal{A}^*$, we have $C_{min}(\mathcal{A}, m) \leq C_{min}(\mathcal{A}^*, m)$ because the minimum of a larger set (e.g., min({1,3,5})) cannot be higher than its subset (e.g., min({3,5})). Second, C_{min} can incorporate any measure m for two items and thus can accommodate different tasks (e.g., mining positive and negative correlations) and be customized to achieve good mining quality.

Leveraging the "apriori" feature of C_{min}, we develop Algorithm APRIORICORRMINING (Figure 3) for discovering complex matchings, in the spirit of the classic Apriori algorithm for association mining [1]. That is, we find all the correlated items with length $l + 1$ based on the ones with length l.

With C_{min}, we can directly define positively correlated attributes in group discovery and negatively correlated attribute groups in matching discovery. A set of attributes $\{A_1, ..., A_n\}$ is *positively correlated attributes*, denoted by *PC*, if $C_{min}(\{A_1, ..., A_n\}, m_p) \geq T_p$, where m_p is a measure for positive correlation and T_p is a given threshold. Similarly, a set of attribute groups $\{G_1, ..., G_m\}$ is *negatively correlated attribute groups*, denoted by *NC*, if $C_{min}(\{G_1, ..., G_m\}, m_n) \geq T_n$, where m_n is a measure for negative correlation and T_n is another given threshold.

Algorithm: MATCHINGSELECTION:
Input: Potential complex matchings $\mathcal{M} = \{M_1, ..., M_v\}$,
Measure m_n
Output: Selected complex matchings
begin:
1 $\mathcal{R} \leftarrow \emptyset$ /* selected n-ary complex matchings */
2 **while** $\mathcal{M} \neq \emptyset$
3 /* select the matching ranked the highest */
4 $M_t \leftarrow$ GETMATCHINGRANKFIRST($\mathcal{M}$, m_n)
5 $\mathcal{R} \leftarrow \mathcal{R} \cup \{M_t\}$
6 **for** each $M_j \in \mathcal{M}$
7 /* remove the conflicting part */
8 $M_j \leftarrow M_j - M_t$
9 /* delete M_j if it contains no matching */
10 **if** $|M_j| < 2$ **then** $\mathcal{M} \leftarrow \mathcal{M} - \{M_j\}$
11 **return** $\mathcal{R}$
end

Algorithm: GETMATCHINGRANKFIRST:
Input: Potential complex matchings $\mathcal{M} = \{M_1, ..., M_v\}$,
Measure m_n
Output: The matching with the highest ranking
begin:
1 $M_t \leftarrow M_1$
2 **for** each $M_j \in \mathcal{M}, 2 \leq j \leq v$
3 **if** $s(M_j, m_n) > s(M_t, m_n)$ **then**
4 $M_t \leftarrow M_j$
5 **if** $s(M_j, m_n) = s(M_t, m_n)$ and $M_j \succeq M_t$ **then**
6 $M_t \leftarrow M_j$
7 **return** M_t
end

Figure 4: Algorithm MATCHINGSELECTION.

Algorithm N-ARYSCHEMAMATCHING shows the pseudo code of the complex matching discovery (Figure 3). Line 2 (group discovery) calls APRIORICORRMINING to mine PC. Lines 3-6 add the discovered groups into $\mathcal{S}_\mathcal{I}$. Line 8 (matching discovery) calls APRIORICORRMINING to mine NC. Similar to [1], the time complexity of N-ARYSCHEMAMATCHING is exponential with respect to the number of attributes. But in practice, the execution is quite fast since correlations exist among semantically related attributes, which is far less than arbitrary combination of all attributes.

3.2 Complex Matching Selection

Correlation mining can discover true semantic matchings and, as expected, also false ones due to the existence of coincidental correlations. For instance, in Books domain, the mining result may have both {author} = {first name, last name}, denoted by M_1 and {subject} = {first name, last name}, denoted by M_2. We can see M_1 is correct, while M_2 is not. The reason for having the false matching M_2 is that in the schema data, it happens that subject does not often co-occur with first name and last name.

The existence of false matchings may cause matching conflicts. For instance, M_1 and M_2 conflict in that if one of them is correct, the other one will not. Otherwise, we get a wrong matching {author} = {subject} by the transitivity of synonym relationship. Since our mining algorithm does not discover {author} = {subject}, M_1 and M_2 cannot be both correct.

Leveraging the conflicts, we select the most confident and consistent matchings to remove the false ones. Intuitively, between conflicting matchings, we want to select the more negatively correlated one because it indicates higher confidence to be real synonyms. For example, our experiment shows that, as M_2 is coincidental, it is indeed that $m_n(M_1) > m_n(M_2)$, and thus we select M_1 and remove M_2. Note that, with larger data size, semantically

	A_p	$\neg A_p$	
A_q	f_{11}	f_{10}	f_{1+}
$\neg A_q$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	f_{++}

Figure 5: Contingency table for test of correlation.

correct matching is more possible to be the winner. The reason is that, with larger size of sampling, the correct matchings are still negatively correlated while the false ones will remain coincidental and not as strong.

Before presenting the selection algorithm, we need to develop a strategy for *ranking* the discovered matchings. That is, for any n-ary complex matching M_j: $G_{j_1} = G_{j_2} = ... = G_{j_w}$, we have a score function $s(M_j, m_n)$ to evaluate M_j under measure m_n.

While Section 3.1 discussed a measure for "qualifying" candidates, we now need to develop another "ranking" measure as the score function. Since ranking and qualification are different problems and thus require different properties, we cannot apply the measure C_{min} (Equation 1) for ranking. Specifically, the goal of qualification is to ensure the correlations passing some threshold. It is desirable for the measure to support downward closure to enable an "apriori" algorithm. In contrast, the goal of ranking is to compare the strength of correlations. The downward closure enforces, by definition, that a larger item set is always less correlated than its subsets, which is inappropriate for ranking correlations of different sizes. Hence, we develop another measure C_{max}, the maximal m_n value among pairs of groups in a matching, as the score function s. Formally,

$$C_{max}(M_j, m_n) = \max m_n(G_{j_r}, G_{j_t}), \forall G_{j_r}, G_{j_t}, j_r \neq j_t. \quad (2)$$

It is possible to get ties if only considering the C_{max} value; we thus develop a natural strategy for tie breaking. We take a "top-k" approach so that s in fact is a set of sorted scores. Specifically, given matchings M_j and M_k, if $C_{max}(M_j, m_n) = C_{max}(M_k, m_n)$, we further compare their second highest m_n values to break the tie. If the second highest values are also equal, we compare the third highest ones and so on, until breaking the tie.

If two matchings are still tie after the "top-k" comparison, we choose the one with richer semantic information. We consider matching M_j *semantically subsumes* matching M_k, denoted by $M_j \succeq M_k$, if all the semantic relationships in M_k are covered in M_j. For instance, {arrival city} = {destination} = {to} $\succeq$ {arrival city} = {destination} because the synonym relationship in the second matching is subsumed in the first one. Also, {author} = {first name, last name} $\succeq$ {author} = {first name} because the synonym relationship in the second matching is part of the first.

Combining the score function and the semantic subsumption, we rank matchings with following rules: 1) If $s(M_j, m_n) > s(M_k, m_n)$, M_j is ranked higher than M_k. 2) If $s(M_j, m_n) = s(M_k, m_n)$ and $M_j \succeq M_k$, M_j is ranked higher than M_k. 3) Otherwise, we rank M_j and M_k arbitrarily. Algorithm GETMATCHINGRANK-FIRST (Figure 4) illustrates the pseudo code of choosing the highest ranked matching with this strategy.

Algorithm MATCHINGSELECTION shows the selection algorithm. We apply a greedy selection strategy by choosing the highest ranked matching, M_t, in each iteration. After choosing M_t, we remove the inconsistent parts in remaining matchings (lines 6 - 10). The final output is the selected n-ary complex matchings without conflict. Note that we need to do the ranking in each iteration instead of sorting all the matchings in the beginning because after removing the conflicting parts, the ranking may change. The time complexity of Algorithm MATCHINGSELECTION is $O(v^2)$, where v is the number of matchings in $\mathcal{M}$.

Example 1: Assume running N-ARYSCHEMAMATCHING in Books domain finds matchings $\mathcal{M}$ as (matchings are followed by their scores):

M_1: {author} = {last name, first name}, 0.95
M_2: {author} = {last name}, 0.95
M_3: {subject} = {category}, 0.92
M_4: {author} = {first name}, 0.90
M_5: {subject} = {last name, first name} , 0.88
M_6: {subject} = {last name}, 0.88 and
M_7: {subject} = {first name}, 0.86.

In the first iteration, M_1 is ranked the highest and thus selected. In particular, although $s(M_1, m_n) = s(M_2, m_n)$, M_1 is ranked higher since $M_1 \succeq M_2$. Now we remove the conflicting parts of the other matchings. For instance, M_2 conflicts with M_1 on author. After removing author, M_2 only contains one attribute and is not a matching any more. So we remove M_2 from $\mathcal{M}$. Similarly, M_4 and M_5 are also removed. The remaining matchings are M_3, M_6 and M_7. In the second iteration, M_3 is ranked the highest and thus also selected. M_6 and M_7 are removed because they conflict with M_3. Now $\mathcal{M}$ is empty and the algorithm stops. The final output is thus M_1 and M_3. ■

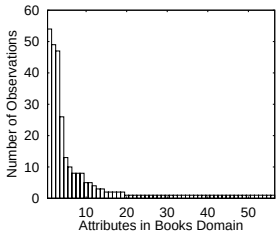


Figure 6: Attribute frequencies in Books domain.

4. CORRELATION MEASURE

In this section, we discuss the positive measure m_p and the negative measure m_n, used as the component of C_{min} (Equation 1) for positive and negative correlation mining respectively in Algorithm N-ARYSCHEMAMATCHING (Section 3).

As discussed in [20], a correlation measure by definition is a testing on the *contingency table*. Specifically, given a set of schemas and two specified attributes A_p and A_q, there are four possible combinations of A_p and A_q in one schema S_i: A_p, A_q are co-present in S_i, only A_p presents in S_i, only A_q presents in S_i, and A_p, A_q are co-absent in S_i. The *contingency table* [5] of A_p and A_q contains the number of occurrences of each situation, as Figure 5 shows. In particular, f_{11} corresponds to the number of co-presence of A_p and A_q; f_{10}, f_{01} and f_{00} are denoted similarly. f_{+1} is the sum of f_{11} and f_{01}; f_{+0}, f_{0+} and f_{1+} are denoted similarly. f_{++} is the sum of f_{11}, f_{10}, f_{01} and f_{00}.

The design of a correlation measure is often empirical. To our knowledge, there is no good correlation measure universally agreed upon [20]. For our matching task, in principle *any* measure can be applied. However, since the semantic correctness of the mining result is of special importance for the schema matching task, we care more the ability of the measures on differentiating various correlation situations, especially the subtlety of negative correlations, which has not been extensively studied before.

We first identify the quality requirements of measures, which are imperative for schema matching, based on the characteristics of Web query interfaces. Specifically, we observe that, in Web interfaces, attribute frequencies are extremely non-uniform, similar to the use of English words, showing some Zipf-like distribution. For instance, Figure 6 shows the attribute frequencies in Books domain: First, the non-frequent attributes results in the sparseness of the schema data (e.g., there are over 50 attributes in Books domain, but each schema only has 5 in average). Second, many attributes are

	A_p	$\neg A_p$	
A_q	5	5	10
$\neg A_q$	5	85	90
	10	90	100

(a1) Example of sparseness problem with measure *Lift*: Less positive correlation but a higher *Lift* = 17.

	A_p	$\neg A_p$	
A_q	1	49	50
$\neg A_q$	1	1	2
	2	50	52

(b1) Example of rare attribute problem with measure *Jaccard*: A_p as rare attribute and *Jaccard* = 0.02.

	A_p	$\neg A_p$	
A_q	81	9	90
$\neg A_q$	9	1	10
	90	10	100

(c1) Example of frequent attribute problem with measure *Jaccard*: A_p and A_q are independent but a higher *Jaccard* = 0.82.

	A_p	$\neg A_p$	
A_q	55	20	75
$\neg A_q$	20	5	25
	75	25	100

(a2) Example of sparseness problem with measure *Lift*: More positive correlation but a lower *Lift* = 0.69.

	A_p	$\neg A_p$	
A_q	1	25	26
$\neg A_q$	25	1	26
	26	26	52

(b2) Example of rare attribute problem with measure *Jaccard*: no rare attribute and *Jaccard* = 0.02.

	A_p	$\neg A_p$	
A_q	8	1	9
$\neg A_q$	1	90	91
	9	91	100

(c2) Example of frequent attribute problem with measure *Jaccard*: A_p and A_q are positively correlated but a lower *Jaccard* = 0.8.

Figure 7: Examples of the three problems.

rarely used, occurring only once in the schemas. Third, there exist some highly frequent attributes, occurring in almost every schema.

These three observations indicate that, as the quality requirements, the chosen measures should be robust against the following problems: *sparseness problem* for both positive and negative correlations, *rare attribute problem* for negative correlations, and *frequent attribute problem* for positive correlations. In this section, we discuss each of them in details.

The Sparseness Problem

In schema matching, it is more interesting to measure whether attributes are often co-present (i.e., f_{11}) or cross-present (i.e., f_{10} and f_{01}) than whether they are co-absent (i.e., f_{00}). Many correlation measures, such as χ^2 and *Lift*, include the count of co-absence in their formulas. This may not be good for our matching task, because the sparseness of schema data may "exaggerate" the effect of co-absence. This problem has also been noticed by recent correlation mining work such as [20, 16, 14]. In [20], the authors use the *null invariance* property to evaluate whether a measure is sensitive to co-absence. The measures for our matching task should satisfy this null invariance property.

Example 2: Figure 7(a) illustrates the sparseness problem with an example. In this example, we choose a common measure: the *Lift* (i.e, *Lift* = $\frac{f_{00}f_{11}}{f_{10}f_{01}}$). (Other measures considering f_{00} have similar behavior.) The value of *Lift* is between 0 to $+\infty$. $Lift = 1$ means independent attributes, $Lift > 1$ positive correlation and $Lift < 1$ negative correlation. Figure 7(a) shows that *Lift* may give a higher value to less positively correlated attributes. In the scenario of schema matching, the table in Figure 7(a2) should be more positively correlated than the one in Figure 7(a1) because in Figure 7(a2), the co-presence (f_{11}) is more frequently observed than the cross-presence (either f_{10} or f_{01}), while in Figure 7(a1), the co-presence has the same number of observations as the cross-presence. However, *Lift* cannot reflect such preference. In particular, Figure 7(a1) gets a much higher *Lift* and Figure 7(a2) is even evaluated as not positively correlated. Similar example can also be found for negative correlation with *Lift*. The reason *Lift* gives an inappropriate answer is that f_{00} incorrectly affects the result. ■

We explored the 21 measures in [20] and the χ^2 measure in [4]. Most of these measures (including χ^2 and *Lift*) suffer the sparseness problem. That is, they consider both co-presence and co-absence in the evaluation and thus do not satisfy the null invariance property. The only three measures supporting the null invariance property are *Confidence*, *Jaccard* and *Cosine*.

The Rare Attribute Problem for Negative Correlation

Although *Confidence*, *Jaccard* and *Cosine* satisfy the null invariance property, they are not robust for the rare attribute problem, when considering negative correlations. Specifically, the rare attribute problem can be stated as when either A_p or A_q is rarely observed, the measure should not consider A_p and A_q as highly negatively correlated because the number of observations is not convincing to make such judgement. For instance, consider the *Jaccard* (i.e., *Jaccard* = $\frac{f_{11}}{f_{11}+f_{10}+f_{01}}$) measure, it will stay unchanged when both f_{11} and $f_{10} + f_{01}$ are fixed. Therefore, to some degree, *Jaccard* cannot differentiate the subtlety of correlations (e.g., f_{10} = 49, f_{01} = 1 and f_{10} = 25, f_{01} = 25), as Example 3 illustrates. Other measures such as *Confidence* and *Cosine* have similar problem. This problem is not critical for positive correlation, since attributes with strong positive correlations cannot be rare.

Example 3: Figure 7(b) illustrates the rare attribute problem. In this example, we choose a common measure: the *Jaccard*. The value of *Jaccard* is between 0 to 1. *Jaccard* close to 0 means negative correlation and *Jaccard* close to 1 positive correlation. Figure 7(b) shows that *Jaccard* may not be able to distinguish the situations of rare attribute. In particular, Jaccard considers the situations in Figure 7(b1) and Figure 7(b2) as the same. But Figure 7(b2) is more negatively correlated than Figure 7(b1) because A_p in Figure 7(b1) is more like a rare event than true negative correlation. ■

To differentiate the subtlety of negative correlations, we develop a new measure, H-measure (Equation 3), as the negative correlation m_n. The value of H is in the range from 0 to 1. An H value close to 0 denotes a high degree of positive correlation; an H value close to 1 denotes a high degree of negative correlation.

$$m_n(A_p, A_q) = H(A_p, A_q) = \frac{f_{01}f_{10}}{f_{+1}f_{1+}}. \quad (3)$$

H-measure satisfied the quality requirements: On one hand, similar to *Jaccard, Cosine* and *Confidence*, H-measure satisfies the null invariance property and thus avoids the sparseness problem by ignoring f_{00}. On the other hand, by multiplying individual effect of f_{01} (i.e., $\frac{f_{01}}{f_{+1}}$) and f_{10} (i.e., $\frac{f_{10}}{f_{1+}}$), H-measure is more capable of reflecting subtle variation of negative correlations.

The Frequent Attribute Problem for Positive Correlation

For positive correlations, we find that *Confidence*, *Jaccard*, *Cosine* and H-measure are not quite different in discovering attribute groups. However, all of them suffer the frequent attribute problem. This problem seems to be essential for these measures: Although they avoid the sparseness problem by ignoring f_{00}, as the cost, they lose the ability to differentiate highly frequent attributes from really correlated ones. Specifically, highly frequent attributes may co-occur in most schemas just because they are so frequently used, not because they have grouping relationship (e.g., In Books domain, isbn and title are often co-present because they are both

very frequently used). This phenomenon may generate uninteresting groups (i.e., *false positives*) in group discovery.

Example 4: Figure 7(c) illustrates the frequent attribute problem with an example, where we still use *Jaccard* as the measure. Figure 7(c) shows that *Jaccard* may give a higher value to independent attributes. In Figure 7(c1), A_p and A_q are independent and both of them have the probabilities 0.9 to be observed; while, in Figure 7(c2), A_p and A_q are really positively correlated. However, *Jaccard* considers Figure 7(c1) as more positively correlated than Figure 7(c2). In our matching task, a measure should not give a high value for frequently observed but independent attributes. ■

The characteristic of false groupings is that the f_{11} value is very high (since both attributes are frequent). Based on this characteristic, we add another measure $\frac{f_{11}}{f_{++}}$ in m_p to filter out false groupings. Specifically, we define the positive correlation measure m_p as:

$$m_p(A_p, A_q) = \begin{cases} 1 - H(A_p, A_q), & \frac{f_{11}}{f_{++}} < T_d \\ 0, & otherwise, \end{cases} \quad (4)$$

where T_d is a threshold to filter out false groupings. To be consistent with m_n, we also use the H-measure in m_p.

5. DATA PREPARATION

The query schemas in Web interfaces are not readily minable in HTML format; as preprocessing, data preparation is essential to prepare "schema transactions" for mining. As shown in Figure 2, data preparation consists of: 1) *form extraction* – extracting attribute entities from query interfaces in Web pages, 2) *type recognition* – recognizing the types of the attribute entities from domain values, and 3) *syntactic merging* – syntactically merging these attribute entities.

Form extraction reads a Web page with query forms and extracts the attribute entities containing attribute names and domains. For instance, the attribute about title in Figure 1(c) is extracted as ⟨name = "title of book", domain = any⟩, where "domain = any" means any value is possible. This task is itself a challenging and independent problem. We solved this problem by a parsing approach with the hypothesis of the existence of hidden syntax [21]. Note that there is no data cleaning in this step and thus the attribute names and domains are raw data.

After extracting the forms, we perform some standard normalization on the extracted names and domains. We first stem attribute names and domain values using the standard Porter stemming algorithm [17]. Next, we normalize irregular nouns and verbs (e.g., "children" to "child," "colour" to "color"). Last, we remove common stop words by a manually built stop word list, which contains words common in English, in Web search (e.g., "search", "page"), and in the respective domain of interest (e.g., "book", "movie").

We then perform type recognition to identify attribute types. As Section 5.1 discusses, type information helps to identify homonyms (i.e., two attributes may have the same name but different types) and constrain syntactic merging and correlation-based matching (i.e., only attributes with compatible types can be merged or matched). Since the type information is not declared in Web interfaces, we develop a *type recognizer* to recognize types from domain values.

Finally, we merge attribute entities by measuring the syntactic similarity of attribute names and domain values (e.g., we merge "title of book" to "title" by name similarity). It is a common data cleaning technique to merge syntactically similar entities by exploring linguistic similarities. Section 5.2 discusses our merging strategy.

5.1 Type Recognition

While attribute names can distinguish different attribute entities, the names alone sometimes lead to the problem of homonyms (i.e., the same name with different meanings) – we address this problem by distinguishing entities by both names and types. For instance, the attribute name departing in the Airfares domain may have two meanings: a datetime type as departing date, or a string type as departing city. With type recognition, we can recognize that there are two different types of departing: departing (datetime) and departing (string), which indicate two attribute entities.

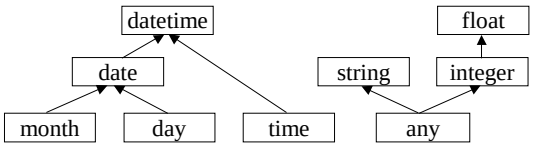


Figure 8: The compatibility of types.

In general, type information, as a constraint, can help the subsequent steps of syntactic merging and correlation-based matching. In particular, if the types of two attributes are not compatible, we consider they denote different attribute entities and thus neither merge them nor match them.

Since type information is not explicitly declared in Web interfaces, we develop a *type recognizer* to recognize types from domain values of attribute entities. For example, a list of integer values denotes an integer type. In the current implementation, type recognition supports the following types: any, string, integer, float, month, day, date, time and datetime. (An attribute with only an input box is given an any type since the input box can accept data with different types such as string or integer.) Two types are *compatible* if one can subsume another (i.e., the *is-a* relationship). For instance, date and datetime are compatible since date subsumes datetime. Figure 8 lists the compatibility of all the types in our implementation.

5.2 Syntactic Merging

We clean the schemas by merging syntactically similar attribute entities, a common data cleaning technique to identify unique entities [8]. Specifically, we develop *name-based merging* and *domain-based merging* by measuring the syntactic similarity of attribute names and domains respectively. Syntactic merging increases the observations of attribute entities, which can improve the effect of correlation evaluation.

Name-based Merging: We merge two attribute entities if they are similar in names. We observe that the majority of deep Web sources are consistent on some concise "core" attribute names (e.g., "title") and others are variation of the core ones (e.g., "title of book"). Therefore, we consider attribute A_p is *name-similar* to attribute A_q if A_p's name $\supseteq$ A_q's name (i.e., A_p is a variation of A_q) and A_q is more frequently used than A_p (i.e., A_q is the majority). This frequency-based strategy helps avoid false positives. For instance, in Books domain, last name is not merged to name because last name is more popular than name and thus we consider them as different entities.

Domain-based Merging: We then merge two attribute entities if they are similar in domain values. In particular, we only consider attributes with string types, since it is unclear how useful it is to measure the domain similarity of other types. For instance, in Airfares domain, the integer values of passengers and connections are quite similar, although they denote different meanings.

We view domain values as a bag of words (i.e., counting the word frequencies). We name such a bag *aggregate values*, denoted as V_A for attribute A. Given a word w, we denote $V_A(w)$ as the frequency of w in V_A. The domain similarity of attributes A_p and A_q is thus the similarity of V_{A_p} and V_{A_q}. In principle, any reasonable similarity function is applicable here. In particular, we choose $sim(A_p, A_q) = \frac{\forall w \in V_{A_p} \cap V_{A_q}, V_{A_p}(w) + V_{A_q}(w)}{\forall w \in V_{A_p} \cup V_{A_q}, V_{A_p}(w) + V_{A_q}(w)}$.

The above three steps, form extraction, type recognition and syntactic merging, clean the schema data as transactions to be mined. More detailed discussion about these data cleaning steps can be found at the extended report [11].

6. EXPERIMENTS

We choose two datasets, TEL-8 dataset and BAMM dataset, of the UIUC Web integration repository [7] as the testbed of the DCM framework. The TEL-8 dataset contains raw Web pages over 447 deep Web sources in 8 popular domains. Each domain has about 20-70 sources. The BAMM dataset contains manually extracted attribute names over 211 sources in 4 domains (with around 50 sources per domain), which was first used by [10].

In the experiment, we assume a perfect form extractor to extract all the interfaces in the TEL-8 dataset into query capabilities by manually doing the form extraction step. The reason we do not apply the work in [21] is that we want to isolate the mining process to study and thus fairly evaluate the matching performance. After extracting the raw data, we do the data cleaning according to the process explained in Section 5. Then, we run the correlation mining algorithm on the prepared data in each domain. Also, in the results, we use attribute name and type together as the attribute identifier for an attribute entity since we incorporate type recognition in data preparation to identify homonyms (Section 5).

To evaluate the matching performance and the H-measure, we extensively test the DCM framework on the two datasets. First, we test our approach on the TEL-8 dataset and the result shows good *target accuracy*. Second, we compare the DCM framework with the MGS framework [10], which also matches Web interfaces by a statistical approach, on its BAMM dataset. The result shows that DCM is empirically close to MGS in discovering simple matchings and further DCM can find complex matchings, which is not supported by MGS. Third, we compare the H-measure with other measures on the TEL-8 dataset and the result shows that H-measure outperforms the others in most cases.

6.1 Metrics

We compare experimentally discovered matchings, denoted by $\mathcal{M}_h$, with correct matchings written by human experts, denoted by $\mathcal{M}_c$. In particular, we adopt the *target accuracy*, a metric initially developed in [10], by customizing the *target questions* to the complex matching scenario. The idea of the target accuracy is to measure how accurately that the discovered matchings answer the target questions. Specifically, for our complex matching task, we consider the target question as, given any attribute, to find its synonyms (i.e., word with exactly the same meaning as another word, e.g., subject is a synonym of category in Books domain), hyponyms (i.e., word of more specific meaning, e.g., last name is a hyponym of author) and hypernyms (i.e., word with a broader meaning, e.g, author is a hypernym of last name).

It is quite complicated to use different measures for different semantic relationships, we therefore report an aggregate measure for simplicity and, at the same time, still reflecting semantic differences. For our discussion here, we name synonym, hyponym and hypernym together as *closenym* – meaning that they all denote some degrees of closeness in semantic meanings. Our target question now is to ask the set of closenyms of a given attribute.

Example 5: For instance, for matching {A} = {B, C}, the closenym sets of attributes A, B, C are {B, C}, {A}, {A} respectively. In particular, the closenym sets of B does not have C since B and C only have grouping relationship. In contrast, for matching {A} = {B} = {C}, the closenym sets of attributes A, B, C are {B, C}, {A, C}, {A, C} respectively. We can see that the difference of matchings can be reflected in the corresponding closenym sets. ■

Except this difference in target question, we use the same metric of target accuracy as [10]. Specifically, we assume a "random querier" to ask for closenym set of each attribute according to the its frequency. The answer to each question is closenym set of the queried attribute in discovered matchings. We define $Cls(A_j|\mathcal{M})$ as the closenym set of attribute A_j. Given $\mathcal{M}_c$ and $\mathcal{M}_h$, the precision and recall of the closenym sets of attribute A_j are:

$$P_{A_j}(\mathcal{M}_h, \mathcal{M}_c) = \frac{|Cls(A_j|\mathcal{M}_c) \cap Cls(A_j|\mathcal{M}_h)|}{|Cls(A_j|\mathcal{M}_h)|} \text{ and}$$

$$R_{A_j}(\mathcal{M}_h, \mathcal{M}_c) = \frac{|Cls(A_j|\mathcal{M}_c) \cap Cls(A_j|\mathcal{M}_h)|}{|Cls(A_j|\mathcal{M}_c)|}.$$

Since more frequently used attributes have higher probabilities to be asked in this "random querier," we compute the weighted average of all the P_{A_j}'s and R_{A_j}'s as the *target precision* and *target recall*. The weight is assigned as $\frac{O_j}{\sum O_k}$, where O_j is the frequency of attribute A_j in the dataset (i.e., its number of occurrences in different schemas). Therefore, *target precision* and *target recall* of $\mathcal{M}_h$ with respect to $\mathcal{M}_c$ are:

$$P_T(\mathcal{M}_h, \mathcal{M}_c) = \sum_{A_j} \frac{O_j}{\sum O_k} P_{A_j}(\mathcal{M}_h, \mathcal{M}_c)$$

$$R_T(\mathcal{M}_h, \mathcal{M}_c) = \sum_{A_j} \frac{O_j}{\sum O_k} R_{A_j}(\mathcal{M}_h, \mathcal{M}_c).$$

6.2 Experimental Results

To illustrate the effectiveness of the mining approach, we only list and count the "semantically difficult" matchings discovered by the correlation mining algorithm, not the simple matchings by the syntactic merging in the data preparation (e.g., {title of book} to {title}). Our experiment shows that many frequently observed matchings are in fact "semantically difficult" and thus cannot be found by syntactic merging.

Result on the TEL-8 Dataset: In this experiment, we run our algorithm (with H-measure as the correlation measure) on the TEL-8 dataset. We set the thresholds T_p to 0.85 and T_d to 0.6 for positive correlation mining and T_n to 0.75 for negative correlation mining. We empirically get these numbers by testing the algorithm with various thresholds and choose the best one. As Section 8 will discuss, more systematic study can be investigated in choosing appropriate threshold values.

Figure 9 illustrates the detailed results of n-ary complex matchings discovered in Books domain. The step of group discovery found 5 likely groups (G_1 to G_5 in Figure 9). In particular, m_p gives a high value (actually the highest value) for the group of last name (any) and first name (any). The matching discovery found 6 likely complex matching (M_1 to M_6 in Figure 9). We can see that M_1 and M_3 are fully correct matchings, while others are partially correct or incorrect. Last, the matching selection will choose M_1 and M_3 (i.e., the correct ones) as the final output.

Figure 10 shows the results on Airfares and Movies. (The results of other domains can be found at the extended report [11]). The third column denotes the correctness of the matching. In particular, Y means a fully correct matching, P a partially correct one and N an incorrect one. Our mining algorithm does find interesting matchings in almost every domain. For instance, in Airfares domain, we find 5 fully correct matchings, e.g., {destination (string)} = {to (string)} = {arrival city (string)}. Also, {passenger (integer)} = {adult (integer), child (integer), infant (integer)} is partially correct because it misses senior (integer).

Since, as a statistical method, our approach replies on "sufficient observations" of attribute occurrences, it is likely to perform more favorably for frequent attributes (i.e., the head-ranked attributes in Figure 6). To quantify this "observation" factor, we report the target accuracy with respect to the attribute frequencies. In particular, we consider the attributes above a *frequency threshold* T (i.e., the number of occurrences of the attribute over the total number of schemas

Step	*Value of*	*Result*	C_{min}	C_{max}
group discovery	$\mathcal{G}$	G_1 = {last name (unknown), first name (any)}	0.94	
		G_2 = {title (any), keyword (any)}	0.93	
		G_3 = {last name (any), title (any)}	0.91	
		G_4 = {first name (any), catalog (any)}	0.90	
		G_5 = {first name (any), keyword (any)}	0.87	
matching discovery	$\mathcal{M}$	M_1: {author (any)} = {last name (any), first name (any)}	0.87	0.87
		M_2: {author (any)} = {last name (any)}	0.87	0.87
		M_3: {subject (string)} = {category (string)}	0.83	0.83
		M_4: {author (any)} = {last name (any), catalog (any)}	0.82	0.82
		M_5: {author (any)} = {first name (any)}	0.82	0.82
		M_6: {category (string)} = {publisher (string)}	0.76	0.76
matching selection	$\mathcal{R}$	R_1: {author (any)} = {last name (any), first name (any)}		0.87
		R_2: {subject (string)} = {category (string)}		0.83

Figure 9: Running Algorithms N-ARYSCHEMAMATCHING and MATCHINGSELECTION on Books domain.

Domain	*Final Output After Matching Selection*	*Correct?*
Airfares	{destination (string)} = {to (string)} = {arrival city (string)}	Y
	{departure date (datetime)} = {depart (datetime)}	Y
	{passenger (integer)} = {adult (integer), child (integer), infant (integer)}	P
	{from (string), to (string)} = {departure city (string), arrival city (string)}	Y
	{from (string)} = {depart (string)}	Y
	{return date (datetime)} = {return (datetime)}	Y
Movies	{artist (any)} = {actor (any)} = {star (any)}	Y
	{genre (string)} = {category (string)}	Y
	{cast & crew (any)} = {actor (any), director (any)}	Y

Figure 10: Experimental results for Airfares and Movies.

Domain	P_T (20%)	R_T (20%)	P_T (10%)	R_T (10%)
Books	1	1	1	1
Airfares	1	1	1	0.71
Movies	1	1	1	1
MusicRecords	1	1	0.76	1
Hotels	0.86	1	0.86	0.87
CarRentals	0.72	1	0.72	0.60
Jobs	1	0.86	0.78	0.87
Automobiles	1	1	0.93	1

Figure 11: Target accuracy of 8 domains.

Domain	$P_T(H)$ (10%)	$R_T(H)$ (10%)	$P_T(\zeta)$ (10%)	$R_T(\zeta)$ (10%)
Books	1	1	0.80	1
Airfares	1	0.71	0.79	0.61
Movies	1	1	0.93	1
MusicRecords	0.76	1	0.76	1
Hotels	0.86	0.87	0.44	0.95
CarRentals	0.72	0.60	0.68	0.62
Jobs	0.78	0.87	0.64	0.87
Automobiles	0.93	1	0.78	1

Figure 12: Comparison of H-measure and *Jaccard*.

is above T) in both discovered matchings and correct matchings to measure the target accuracy. Specifically, we run the algorithms on all the attributes and then report the target accuracy in terms of the frequency-divided attributes. In the experiment, we choose T as 20% and 10%.

Consider the Airfares domain, if we only consider the attributes above 20% frequency in the matching result, only 12 attributes are above that threshold. The discovered matchings in Figure 10 become {destination (string)} = {to (string)}, {departure date (datetime)} = {depart (datetime)}, and {return date (datetime) = return (datetime)}. (The other attributes are removed since they are all below 20% frequency.) These three matchings are exactly the correct matchings the human expert can recognize among the 12 attributes and thus we get 1.0 in both target precision and recall.

Next, we apply the 10% frequency threshold and get 22 attributes. The discovered matchings in Figure 10 are unchanged since all the attributes (in the matchings) are now passing the threshold. Compared with the correct matchings among the 22 attributes, we do miss some matchings such as {cabin (string)} = {class (string)} and {departure (datetime) = departure date (datetime)}. Also, some matchings are partially correct such as {passenger (integer)} = {adult (integer), child (integer), infant (integer)}. Hence, we get 1.0 in target precision and 0.71 in target recall.

Figure 11 lists the target accuracies of the 8 domains under thresholds 20% and 10%. From the result, we can see that our approach does perform better for frequent attributes.

Result on the BAMM Dataset: We test the DCM framework on the BAMM dataset used in [10]; the result shows that DCM is empirically close to the MGS framework in [10] on discovering simple 1:1 matchings and further we can find complex matchings that MGS cannot. Since the BAMM dataset only contains manually extracted attribute names, we skip the data preparation step in this experiment. The result shows that we can discover almost all the simple 1:1 matchings found by MGS. In particular, we find {subject} = {category} in Books, {style} = {type} = {category} in Automobiles, {actor} = {artist} and {genre} = {category} in Movies, and {album} = {title} and {band} = {artist} in MusicRecords. Further, DCM can find the complex matchings {author} = {last name, first name} in Books, while MGS can only find either {author} = {last name} or {author} = {first name}.

Evaluating the H-Measure: We compare the H-measure with other measures and the result shows that H-measure gets better target accuracy. As an example, we choose *Jaccard* (ζ) as the measure we compare to. With *Jaccard*, we define the m_p and m_n as

$$m_p(A_p, A_q) = \begin{cases} \zeta(A_p, A_q), & \frac{f_{11}}{f_{++}} < T_d \\ 0, & otherwise, \end{cases}$$

and

$$m_n(A_p, A_q) \quad = \quad 1 - \zeta(A_p, A_q).$$

We set the T_p and T_n for this *Jaccard*-based m_p and m_n as 0.5 and 0.9 respectively. We compare the target accuracy of H-measure and *Jaccard* in the situation of 10% frequency threshold. The result (Figure 12) shows that H-measure is better in both target precision and target recall in most cases. Similar comparisons show that H-measure is also better than other measures such as *Cosine* and *Confidence*.

7. RELATED WORK

Schema matching is important for schema integration [2, 19] and thus has got great attention. However, existing schema matching works mostly focus on simple 1:1 matching [18, 9, 15] between two schemas. Complex matching has not been extensively studied, mainly due to the much more complex search space of exploring all possible combinations of attributes. Consider two schemas with u and v attributes respectively, while there are only $u \times v$ potential 1:1 matchings, the number of possible m:n matchings is exponential. Also, it is still unclear that how to compare the similarity between two groups of attributes. In contrast, this paper proposes to discover complex matchings by holistically matching all the schemas together. Specifically, we explore the *co-occurrences* information across schemas and develop a *correlation mining* approach.

Unlike previous correlation mining algorithms, which mainly focus on finding strong positive correlations [1, 20, 16, 14, 4], our algorithm cares both positive and negative correlations. In particular, as a new application for correlation mining, the correctness of schema matching mainly depends on the subtlety of negative correlations. We thus study the rare attribute problem and develop the H-measure, which empirically outperforms existing ones on evaluating negative correlations.

Our previous schema matching work, the MGS framework [10], also matches Web interfaces by exploiting holistic information. Although built upon the same insight, DCM is different from MGS in: 1) *abstraction*: DCM abstracts schema matching as correlation mining, while MGS as hidden model discovery by applying statistical hypothesis testing. The difference in abstraction leads to fundamentally different approaches. 2) *expressiveness*: DCM finds m:n matchings, while MGS currently finds 1:1 matchings and it is unclear that how it can be extended to support m:n matchings.

8. CONCLUDING DISCUSSION

In our development of the mining-based matching approach, we also observed several further opportunities and open issues that warrant more investigation. First, it is interesting to know whether our observation and approach can cross the domain boundary. Specifically, given a set of Web interfaces across different domains, we hope to know whether there still are interesting patterns that reveal some semantic relationships among attributes, as we have observed for sources in one domain.

Second, more systematic study can be investigated for choosing appropriate correlation measures and threshold values. In this paper, we choose the H-measure based on the observations of the data and the threshold values according to the empirical experiments. We expect a more formal and systematic study to help the design of measures and the evaluation of threshold values.

Third, to validate and refine the matching results, we may send some trial probings through Web interfaces. For instance, given two online movie sources, one using actor and the other using star, we can send some sample queries on these two sources with same values on actor and star. If they often return overlapping results, we consider the matching {actor} = {star} is correct. However, this probing brings new challenges to solve. In particular, for complex matchings (e.g.,{author} = {last name, first name}), schema composition has to be done before probing, which is itself a difficult problem. Also, it is unclear that how to automatically collect sample queries for each domain. Last, with current techniques, it is difficult to accurately compare the query results in Web pages.

In summary, this paper explores co-occurrence patterns among attribute to tackle the complex matching problem. This exploration is well suited for the integration of large-scale heterogenous data sources, such as the deep Web. Specifically, we abstract complex matching as correlation mining and develop the DCM framework. Further, we propose a new correlation measure, H-measure, for mining negative correlations. Our experiments validate the effectiveness of both the mining approach and the H-measure.

9. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, 1993.

[2] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[3] M. K. Bergman. The deep web: Surfacing hidden value. Technical report, BrightPlanet LLC, Dec. 2000.

[4] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. In *SIGMOD Conference*, 1997.

[5] H. D. Brunk. *An Introduction to Mathematical Statistics*. New York, Blaisdell Pub. Co., 1965.

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

[7] K. C.-C. Chang, B. He, C. Li, and Z. Zhang. The UIUC web integration repository. Computer Science Department, University of Illinois at Urbana-Champaign. http://metaquerier.cs.uiuc.edu/repository, 2003.

[8] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD Conference*, 2003.

[9] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

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

[11] B. He, K. C.-C. Chang, and J. Han. Automatic complex schema matching across web query interfaces: A correlation mining approach. Technical Report UIUCDCS-R-2003-2388, Dept. of Computer Science, UIUC, July 2003.

[12] B. He, T. Tao, and K. C.-C. Chang. Clustering structured web sources: A schema-based, model-differentiation approach. In *EDBT'04 ClustWeb Workshop*, 2004.

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

[14] Y.-K. Lee, W.-Y. Kim, Y. D. Cai, and J. Han. Comine: Efficient mining of correlated patterns. In *Proc. 2003 Int. Conf. Data Mining*, Nov. 2003.

[15] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the 27th VLDB Conference*, pages 49–58, 2001.

[16] E. Omiecinski. Alternative interest measures for mining associations. *IEEE Trans. Knowledge and Data Engineering*, 15:57–69, 2003.

[17] M. Porter. The porter stemming algorithm. Accessible at http://www.tartarus.org/˜martin/PorterStemmer.

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

[19] L. Seligman, A. Rosenthal, P. Lehner, and A. Smith. Data integration: Where does the time go? *Bulletin of the Tech. Committee on Data Engr.*, 25(3), 2002.

[20] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *ACM SIGKDD Conference*, July 2002.

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