

Mining Complex Matchings across Web Query Interfaces

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. As a new attempt, this paper studies such matching as a data mining problem. Specifically, 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, which consists of *dual mining* of positive and negative correlations. We evaluate our approach on deep Web sources in several object domains (e.g., Books and Airfares) and the results show that the correlation mining approach does discover semantically meaningful matchings among attributes.

1. INTRODUCTION

In the 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 medi-

ation across deep Web sources. On this 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*, as Figure 1(a) shows). *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. 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 note 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. Also, our recent work addresses the preceding step of extracting query schemas from Web query interfaces [22].

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 the 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}.

Although 1:1 matching has got great attention [18, 9, 15, 11], *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 2 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 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 ob-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DMKD'04 June 13, 2004, Paris, France.

Copyright 2004 ACM ISBN 1-58113-908-X/04/06...\$5.00.



Figure 1: Examples of “Fragment” Web Query Interfaces.

serve 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 view of the schema matching problem. Specifically, given Web pages containing query interfaces, this paper develops the abstraction of schemas as “transactions” and discovers potential complex matchings by mining positive and negative correlations. Hence, the algorithm consists of automatic *data preparation* and *correlation mining*: First, 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). Second, the correlation mining step, the main focus of this paper, discovers potential complex matchings with dual mining of positive and negative correlations (Section 4).

Note that in this paper, we focus on matching, not *composition*. Matching finds the semantic relationship among source schemas, while composition finds the data translation between sources [21] and is essentially a query reformulation problem. For instance, matching discovers {author} = {last name, first name}, while composition answers how to combine last name and first name to form author and vice versa. Matching itself is important and challenging for schema integration. In this paper, we focus on matching and consider composition as a subsequent task after matching.

To evaluate the matching performance, we test our approach on manually collected deep Web sources in 4 popular domains: Books, Airfares, Movies and MusicRecords. In particular, we collected 50 to 80 sources for each domain. The result shows that our algorithm can find semantically meaningful matchings in every domain (Section 6). Hence, while this work is preliminary, it seems encouraging and promising on developing a mining view for the schema matching problem.

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 summary, this paper builds 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. In our development, we also observed several interesting issues that deserve future investigation: How to analyze and “refine” the mining results to provide more useful and correct answers to users? How to systematically choose a better correlation measure? We discuss these issues in Section 7.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 presents our motivating observations of integrating the deep Web. Section 4 develops the mining algorithms and some other related issues. Section 5 presents the data preparation step. Section 6 reports our case studies. Section 7 discusses the future work and Section 8 concludes this paper.

2. 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 schema matching scenario, as a new application for correlation mining, cares both positive and negative correlations. In particular, we develop the C_{min} measure based on our observations of matchings.

Some recent schema matching works, such as the REVERE proposal [10] and our previous MGS framework [11], also exploit holistic information. REVERE suggests to separately construct a schema corpus as a “knowledge base” to assist matching two unseen sources. Our work is different in that we leverage input schemas themselves as the “corpus” to match all the sources. Our previous schema matching work, the MGS framework, also matches Web interfaces by exploiting holistic information. Although built upon the

same insight, our work is different from MGS in: 1) *abstraction*: we abstract 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*: our work 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.

3. 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 *grouping 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.

We develop a new approach, *correlation mining*, to discover n -ary complex matchings, which consists of two steps: 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 4) discovers n -ary complex matchings by first finding potential attribute groups using positive correlations and then potential complex matchings using negative correlations.

4. COMPLEX MATCHING AS CORRELATION MINING

To transform schema matching into a data mining problem, 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 $S_{\mathcal{I}} = \{S_1, \dots, S_u\}$ in the same domain, where each schema S_i is a transaction of attribute identifiers, find all the matchings $\mathcal{M} = \{M_1, \dots, M_v\}$. Each M_j is an n -ary complex matching $G_{j_1} = G_{j_2} = \dots = G_{j_w}$, where each G_{j_k} is an attribute group and $G_{j_k} \subseteq \bigcup_{t=1}^u S_t$. Semantically, each M_j should represent the synonym relationship of attribute groups G_{j_1}, \dots, G_{j_w} and each G_{j_k} should represent the grouping relationship of attributes in G_{j_k} .*

As Section 1 motivated, we develop a correlation mining algorithm, with respect to the above abstraction. *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

```

Algorithm: N-ARYSCHEMAMATCHING:
Input: InputSchemas  $\mathcal{S}_{\mathcal{I}} = \{S_1, \dots, S_u\}$ ,
      Measures  $m_p, m_n$ , Thresholds  $T_p, T_n$ 
Output: Potential  $n$ -ary complex matchings
begin:
1 /* group discovery */
2  $\mathcal{G} \leftarrow \text{APRIORICORRMINING}(\mathcal{S}_{\mathcal{I}}, m_p, T_p)$ 
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  $\mathcal{M} \leftarrow \text{APRIORICORRMINING}(\mathcal{S}_{\mathcal{I}}, m_n, T_n)$ 
9 return  $\mathcal{M}$ 
end

```

(a) Algorithm N-ARYSCHEMAMATCHING.

```

Algorithm: APRIORICORRMINING:
Input: InputSchemas  $\mathcal{S}_{\mathcal{I}} = \{S_1, \dots, S_u\}$ ,
      Measures  $m$ , Threshold  $T$ 
Output: Correlated items
begin:
1  $X \leftarrow \emptyset$ 
2  $\mathcal{V} \leftarrow \bigcup_{i=1}^u S_i, S_i \in \mathcal{S}_{\mathcal{I}}$ 
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

```

(b) Algorithm APRIORICORRMINING.

Figure 2: Algorithms for Mining Complex Matchings.

(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.

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 $\{\text{last name, first name}\}$ 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 $\{\text{last name}\}$ and $\{\text{first name}\}$ when adding G_{lf} , because G_{lf} is only, at this point, a potential group and may not survive in matching discovery.

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

Correlation mining, at the heart, requires a measure to gauge correlation of a set of n items; our observation indicates “clique”-like 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 $\{\text{destination}\} = \{\text{to}\} = \{\text{arrival city}\}$, negative correlations exist between any two groups). We have similar observation on the attributes with grouping relationships. Motivated by these observations, we design the measure as:

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

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* value of the pairwise evaluation, thus requiring *all* pairs to meet this minimal “strength,” which effectively enforces the “clique”-like correlation.

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 should have a desirable “apriori” property (i.e., downward closure), to develop an efficient algorithm. 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 cannot be higher than its subset. Second, C_{min} can incorporate any pairwise correlation measure m as the basis. Hence, C_{min} can accommodate different tasks (e.g., mining both positive and negative correlations) and be customized to achieve good mining quality.

Leveraging the “apriori” feature of C_{min} , we develop Algorithm APRIORICORRMINING (Figure 2(b)) for discovering complex matchings, in the spirit of the classic Apriori algorithm for association mining [1]. That is, we generate the candidates with length $l + 1$ based on the correlated items with length l , and then prune uncorrected candidates by checking pairwise correlations.

As mentioned earlier, our goal is to develop group discovery for positive correlated attributes and matching discovery for negatively correlated attribute groups. With C_{min} , we can formally define both positively correlated attributes and negatively correlated attribute groups: A set of attributes $\{A_1, \dots, A_n\}$ is positively correlated if $C_{min}(\{A_1, \dots, A_n\}, m_p) \geq T_p$, where m_p is a measure for positive correlation and T_p is a given threshold. Similarly, a set of attribute groups $\{G_1, \dots, G_m\}$ is negatively correlated, if $C_{min}(\{G_1, \dots, G_m\}, m_n) \geq T_n$, where m_n is a measure for negative correlation and T_n is another given threshold.

Algorithm N-ARYSCHEMAMATCHING shows the pseudo code of the complex matching discovery (Figure 2(a)). Line 2 (group discovery) calls APRIORICORRMINING to mine positively correlated attributes. Lines 3-6 add the discovered groups into $\mathcal{S}_{\mathcal{I}}$. Line 8 (matching discovery) calls APRIORICORRMINING to mine negatively correlated attribute groups. 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 only exist among semantically related attributes, which are far less than arbitrary combination of all attributes.

5. DATA PREPARATION

To enable a data mining approach for schema matching, we develop the abstraction of schemas as transactions. Since 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. The data preparation step consists of: 1) *Form extraction* – extracting attribute names and domain values (i.e., the values in the selection list) from query interfaces in Web pages. 2) *Type recognition* – recognizing attribute types (e.g., string type, integer type) from the domain values. These two steps together complete the preparation of attribute entities, which Section 4 required as input to the mining process. 3) *Syntactic merging* – merging syntactically similar attribute entities.

The form extraction step reads a Web page with query forms and extracts attribute names and domain values. 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. To address this problem, we developed an effective parsing approach with the assumption of the existence of “hidden syntax” [22]. Note that there is no data cleaning in this step and thus the attribute names and domains are raw data.

After form extraction, 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”) according to the grammar lists in [5]. 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 will discuss, 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 the traditional linguistic approaches based on textual similarity [18]. Section 5.2 discusses our merging strategy.

Note that syntactic merging, as one step of the preprocessing, can only find very simple correspondences (i.e., syntactically similar ones). However, many “semantically-related” (but syntactically-dissimilar) correspondences exist in ev-

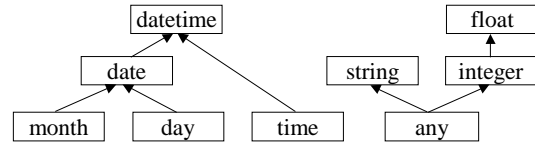


Figure 3: The compatibility of types.

ery domain. For instance, consider the matching {actor} = {star} in Movies. In Web interfaces, both attributes actor and star are followed by only an input box (i.e., there are no domain values). Therefore, the syntactic similarities of both names and domain values cannot discover such matchings. Similar examples can be found in every domain (e.g., {title} = {album} in MusicRecords, {author} = {last name, first name} in Books, {passengers} = {adults, seniors, children, infants} in Airfares). Discovering these “semantically-related” (but syntactically-dissimilar) matchings, the main focus of this paper (Section 4), is more important and difficult.

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: departing (datetime) and departing (string). Although they have the same attribute name, they are distinct attribute entities due to the difference in type.

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 do not merge them in the syntactic merging.

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 various 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 3 lists the compatibility of all the types in our implementation.

5.2 Syntactic Merging

We clean the schemas by merging syntactically similar attribute entities, which is a common data cleaning technique to identify unique entities (e.g., [8]). Specifically, we develop *name-based merging* and *domain-based merging* by measuring the syntactic similarity of attribute names and domains respectively. We note that such syntactic merging increases the number of observations (i.e., the frequencies) of attribute entities, which can improve the effect of correlation evaluation of Section 4.

Name-based Merging: We merge two attribute entities if they are similar in names. We observe that many attribute name variations are syntactically similar by sharing the same “core” terms (e.g., “title” and “title of book” share the term “title”). 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 the Books domain, **last name** is not merged to **name** because **last name** is more popular (i.e., frequent) than **name** and thus we consider them as different entities.

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

We view the domain values of an attribute as a bag of words and thus the words are repeatable in occurrences. 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 $\text{sim}(A_p, A_q) = \frac{\forall w \in V_{A_p} \cap V_{A_q}, V_{A_p}(w) + V_{A_q}(w)}{\forall w \in V_{A_p} \cup V_{A_q}, V_{A_p}(w) + V_{A_q}(w)}$.

The above three steps, form extraction, type recognition and syntactic merging, clean the schema data as transactions to be mined.

6. CASE STUDIES

To evaluate the matching performance, we test our approach on manually collected and extracted deep Web sources in 4 popular domains: Books, Airfares, Movies and MusicRecords. We collected those sources using Web directories (e.g., BrightPlanet.com, Yahoo.com) and search engines (e.g. Google.com). In particular, for each domain, we collected about 50 to 80 sources.

We assume a perfect form extractor to extract all the attribute names and domain values by manually performing the form extraction step. We note that the form extraction techniques develop in [22] can be applied rather effectively for this purpose; however, we assume perfect extraction 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 mining algorithm on the prepared data in each domain. Also, when reporting the results in this section, 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).

We choose *Jaccard* (i.e., $\zeta = \frac{P(A,B)}{P(A)+P(B)-P(A,B)}$) as the measures, m_p and m_n , used in Equation 1. In particular, we let $m_p = \zeta$ and $m_n = 1 - \zeta$. (We choose the popular correlation measure *Jaccard*, which has some good properties [14]).

As Section 7 will discuss, we plan to take a more systematic study on choosing good measures in the future work). Also, we set the thresholds T_p to 0.5 for positive correlation mining and T_n to 0.95 for negative correlation mining. We empirically get these numbers by testing the algorithm with various thresholds and choose the best one. (As Section 7 will discuss, in the future work, we plan to investigate a more systematic study on choosing appropriate threshold values.)

Figure 4 illustrates the detailed results of n -ary complex matchings discovered in the Books domain. The step of group discovery found 5 likely groups (G_1 to G_5 in Figure 4). In particular, the correct semantic grouping of **last name** (any) and **first name** (any) is discovered. The matching discovery found 6 likely complex matchings (M_1 to M_6 in Figure 4). We can see that M_1 and M_6 are fully correct matchings, while others are partially correct (i.e., M_2 and M_3) or incorrect (i.e., M_4 and M_5). As Section 7 will discuss, we can develop strategies such as removing semantically subsumed or incompatible matchings to select the correct matchings among all the potential ones.

Figure 5 shows the results on Airfares, Movies and MusicRecords. The third column denotes the correctness of the matching. Y means a fully correct matching, P a partially correct one and N an incorrect one. Our mining algorithm does find interesting matchings in every domain. For instance, in the Airfares domain, we find 6 fully correct matchings, e.g., {**from** (string), **to** (string)} = {**departure city** (string), **arrival city** (string)}. Also, {**passenger** (integer)} = {**adult** (integer), **child** (integer)} is partially correct because it misses **senior** (integer) and **infant** (integer). Overall, the experiments show that our mining approach, as the initial step of complex matching, is quite effective in discovering potential matchings.

7. FUTURE WORK

In this preliminary work of the mining-based approach for schema matching, while the initial result is promising, we also observed several further opportunities and open issues that warrant more investigation.

First, as a statistical approach, correlation mining can discover nontrivial semantic matchings (beyond the straightforward syntactic similarities) and, as expected, also false ones due to the existence of coincidental correlations. Therefore, to make the mining result “cleaner,” we must further develop a selection strategy to filter out false matchings and partially correct ones. To begin with, while there are many matchings discovered, not all of them are compatible; therefore, we can remove the incompatible ones by their “confidences” of correlation. For instance, consider the discovered matchings in the Books domain (Figure 4), if M_3 is correct, M_4 and M_5 should not be considered as correct due to the violation of the transitivity of synonym relationship. Specifically, if M_3 and M_5 are both correct, because synonym relationship is transitive, we should also discover matching **author** (any) = **ISBN** (any), which is not listed in our result. Therefore, M_5 is *incompatible* with M_3 . Since M_3 has higher confidence to be the correct one, we can remove all the matchings incompatible with M_3 (i.e., M_4 and M_5). Further, M_1 semantically subsumes M_2 and M_3 , since M_2 and M_3 both

Step	Value of	Result	C_{min}
group discovery	\mathcal{G}	$G_1 = \{\text{title (any), ISBN (any), author (any)}\}$	0.79
		$G_2 = \{\text{ISBN (any), author (any)}\}$	0.79
		$G_3 = \{\text{title (any), author (any)}\}$	0.79
		$G_4 = \{\text{title (any), ISBN (any)}\}$	0.74
		$G_5 = \{\text{last name (any), first name (any)}\}$	0.55
matching discovery	\mathcal{M}	$M_1: \{\text{author (any)}\} = \{\text{last name (any), first name (any)}\}$	0.98
		$M_2: \{\text{author (any)}\} = \{\text{last name (any)}\}$	0.98
		$M_3: \{\text{author (any)}\} = \{\text{first name (any)}\}$	0.98
		$M_4: \{\text{first name (any)}\} = \{\text{ISBN (any), author (any)}\}$	0.96
		$M_5: \{\text{first name (any)}\} = \{\text{ISBN (any)}\}$	0.96
		$M_6: \{\text{subject (string)}\} = \{\text{category (string)}\}$	0.96

Figure 4: Running Algorithm N-arySchemaMatching on the Books domain.

Domain	Final Output After Matching Selection	Correct?
Airfares	$\{\text{passenger (integer)}\} = \{\text{adult (integer), child (integer)}\}$	P
	$\{\text{passenger (integer)}\} = \{\text{adult (integer)}\}$	P
	$\{\text{from (string), to (string)}\} = \{\text{departure city (string), arrival city (string)}\}$	Y
	$\{\text{from (string)}\} = \{\text{departure city (string), arrival city (string)}\}$	P
	$\{\text{from (string), to (string)}\} = \{\text{departure city (string)}\}$	P
	$\{\text{from (string)}\} = \{\text{departure city (string)}\}$	Y
	$\{\text{destination (string)}\} = \{\text{to (string)}\}$	Y
	$\{\text{from (string), to (string)}\} = \{\text{arrival city (string)}\}$	P
	$\{\text{from (string)}\} = \{\text{arrival city (string)}\}$	N
	$\{\text{to (string)}\} = \{\text{departure city (string), arrival city (string)}\}$	P
	$\{\text{passenger (integer)}\} = \{\text{child (integer)}\}$	P
	$\{\text{to (string)}\} = \{\text{departure city (string)}\}$	N
	$\{\text{arrival city (string)}\} = \{\text{to (string)}\}$	Y
	$\{\text{departure date (datetime)}\} = \{\text{depart (datetime)}\}$	Y
	$\{\text{departure date (datetime), return date (datetime)}\} = \{\text{departure (datetime)}\}$	P
$\{\text{return date (datetime)}\} = \{\text{departure (datetime)}\}$	N	
$\{\text{departure date (datetime)}\} = \{\text{departure (datetime)}\}$	Y	
$\{\text{passenger (integer)}\} = \{\text{infant (integer)}\}$	P	
Movies	$\{\text{actor (any)}\} = \{\text{star (any)}\}$	Y
	$\{\text{genre (string)}\} = \{\text{category (string)}\}$	Y
	$\{\text{keyword (any)}\} = \{\text{star (any)}\}$	P
MusicRecords	$\{\text{genre (any)}\} = \{\text{studio (any)}\}$	N
	$\{\text{title (any)}\} = \{\text{album (any)}\}$	Y
	$\{\text{keyword (any)}\} = \{\text{song (any), album (any)}\}$	P
	$\{\text{keyword (any)}\} = \{\text{album (any)}\}$	P
	$\{\text{keyword (any)}\} = \{\text{song (any)}\}$	P

Figure 5: Experimental results for Airfares, Movies and MusicRecords.

are only part of M_1 . Such *semantic subsumption* can be considered as another selection strategy and thus we can remove the paretilly correct M_2 and M_3 . Combining these two strategies, we will only output the correct matchings M_1 and M_6 . In summary, we need to formalize these ideas and develop an algorithm that integrates all the selection strategies.

Second, we plan to pursue more systematic study for choosing appropriate correlation measures and threshold values. In this paper, we empirically choose the *Jaccard* measure. However, *Jaccard* may not be the best measure. For instance, consider the discovered groups in Books domain (Figure 4), G_1 to G_4 do not really have grouping relationships. They pass the threshold only because their attributes (e.g., ISBN, title, author) are highly frequent and thus may co-occur in most schemas. In general, we believe it is possible to develop more robust measures to meet to characteristics of schema attributes (e.g., to avoid the problem of highly frequent attributes as false groupings). Also, in the experiments, we set the threshold values T_p and T_n by empirical testing. We plan to take a more systematic study to decide those values (e.g., by evaluating of the influence of different thresholds on the mining results).

Third, as inherent in statistical methods, our approach handles only frequent attributes with sufficient observations; to fix this problem, we can integrate other matching techniques into the mining framework. As our survey of the deep Web [6] observed, the frequent attributes, while small in number, dominate 80% occurrences (because of the Zipf-like distribution). Thus, in the sense of the classic 80-20 rule, our technique is appealing in coping with at least the “vital few,” although it leaves out the “trivial many” (that are likely unimportant). Further, with more sources, more attributes will be sufficiently observed and thus handled by our approach. Finally, we stress that, for large-scale integration of sources for numerous domains on the Web, automatic techniques like ours (even only for frequent attributes in each domain) will be imperative, as manual matching will not scale. In the future work, we will integrate various techniques [18] for schema matching into our mining framework, to build a comprehensive system for matching Web interfaces.

Fourth, to validate and refine the matching results, we may send some trial probings through the Web query 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 query results, we consider the matching $\{\text{actor}\} = \{\text{star}\}$ is correct. However, this probing brings new challenges to solve. In particular, for complex matchings (e.g., $\{\text{author}\} = \{\text{last name, first name}\}$), schema composition has to be done before probing (e.g., compose last name and first name into author), 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.

Fifth, we may collect some content data to help the syntactic merging. In this paper, we only focus on the interface information, because in many settings, such matching must happen before being able to retrieve the content. However, besides Web interfaces, there may be some other means to retrieve the content. For instance, some deep Web sites provide browsing directory to reach their content (e.g., the subject directory of books in *amazon.com*). Therefore, it is possible to collect some content data through these browsing directories to help the syntactic merging. However, as discussed above, it is still unclear, with current techniques, how to efficiently and effectively browse the deep Web and then extract the content.

8. CONCLUSION

In summary, this paper explores co-occurrence patterns among attribute to tackle complex matching as a data mining problem. This exploration is well suited for the integration of large-scale heterogeneous data sources, such as the deep Web. Specifically, we abstract complex matching as correlation mining and develop a dual mining algorithm. Our initial studies show the promise of the correlation mining view of the schema matching problem.

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] P. Byrd. Lists of grammar lists. Accessible at <http://www.gsu.edu/~wwwesl/egw/grlists.htm>.
- [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] A. Halevy, O. Etzioni, A. Doan, Z. Ives, J. Madhavan, L. McDowell, and I. Tatarinov. Crossing the structure chasm. *Conf. on Innovative Database Research*, 2003.
- [11] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *SIGMOD Conference*, 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] L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In *SIGMOD Conference*, 2001.
- [22] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best effort parsing with hidden syntax. In *SIGMOD Conference*, 2004.