Automatic Complex Schema Matching across Web Query Interfaces: A Correlation Mining Approach*

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

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 article 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 preprocessing*, *dual mining* of positive and negative correlations, and finally *matching construction*. We evaluate the DCM framework on manually extracted interfaces and the results show good accuracy for discovering complex matchings. Further, to automate the entire matching process, we incorporate automatic techniques for interface extraction. Executing the DCM framework on automatically extracted interfaces, we find that the inevitable errors in automatic interface extraction may significantly affect the matching result. To make the DCM framework robust against such "noisy" schemas, we integrate it with a novel "ensemble" approach, which creates an ensemble of DCM matchers, by randomizing the schema data into many *trials* and aggregating their ranked results by taking majority voting. As a principled basis, we provide analytic justification of the robustness of the ensemble approach. Empirically, our experiments show that the "ensemblization" indeed significantly boosts the matching accuracy, over automatically extracted and thus noisy schema data. By employing the DCM framework with the ensemble approach, we thus complete an automatic process of matchings Web query interfaces.

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

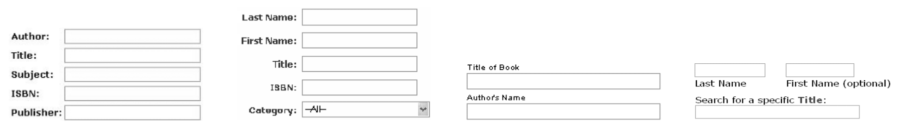
General Terms: Algorithms, Measurement, Experimentation, Performance

Additional Key Words and Phrases: Data integration, schema matching, deep Web, correlation mining, ensemble, bagging predictors

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 [Bergman 2000] estimated that 96,000 "search sites" and 550 billion content pages in this deep Web. Our recent study [Chang et al. 2004] in April 2004 estimated 450,000 online databases. 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



(a) *amazon.com* **(b)** *randomhouse.com* **(c)** *bn.com* **(d)** *1bookstreet.com*

Fig. 1. Example fragments of web query interfaces.

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[1].

In particular, matching Web interfaces in the same domain (*e.g.*, Books, Airfares), the focus of this article, 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 article addresses the problem of discovering matchings among those interfaces. Section 2 will discuss some potential applications of our matching work and the general abstraction of the matching problem. We note that our input, a set of Web pages with interfaces in the same domain, can be either manually collected [Chang et al. 2003] or automatically generated [Ipeirotis et al. 2001].

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 the Airfares domain, $\{$passengers$\}$ = $\{$adults, seniors, children, infants$\}$. Hence, discovering complex matchings is critical to integrating the deep Web.

Although 1:1 matching has got great attention [Rahm and Bernstein 2001; Doan et al. 2001; Madhavan et al. 2001; He and Chang 2003], 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 in isolation, we match *all* the schemas at the same time[2]. 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,

[1]Note that our focus in this article is to match attributes across Web query interfaces. We believe this focus itself is an interesting and challenging problem to study. Some subsequent and related problems, such as how to map data between sources after matching, how to decompose a user's query onto specific sources and how to integrate query result pages from multiple sources, are also interesting topics to study but are beyond the scope of this work. Discussions about putting some of these tasks together as complete systems can be found at [Chang et al. 2005] (*i.e.*, the MetaQuerier system) and [Zhang et al. 2005] (*i.e.*, the Form Assistant system), as Section 2 will briefly discuss.

[2]We note that, in the literature, the term "matching" was originally intended and also naturally suggests correspondences between *two* schemas. In this paper, we take a broader view by considering matching, lacking a better term, as finding semantic correspondences among a set of schemas.

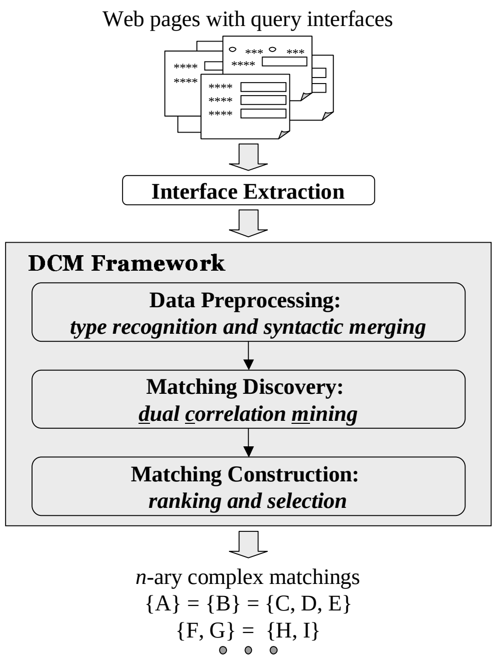

Web pages with query interfaces
Interface Extraction
DCM Framework
Data Preprocessing: type recognition and syntactic merging
Matching Discovery: dual correlation mining
Matching Construction: ranking and selection
n-ary complex matchings
{A} = {B} = {C, D, E}
{F, G} = {H, I}

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

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.

Note that 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. For instance, in Airfares domain, we may have the matching {adults, seniors, children, infants} = {passengers} = {number of tickets}, which is a 4:1:1 matching. We name this type of matching *n-ary complex matching*, which can be viewed as a composition of several binary m:n matchings.

These observations motivate us to develop a correlation mining abstraction of the schema matching problem. Specifically, given extracted schemas from Web query interfaces, this article develops a streamlined process, the DCM framework, for mining complex matchings, consisting of *data preprocessing*, *matching discovery* and *matching construction*, as Figure 2 shows. Since the query schemas in Web interfaces are not readily minable in HTML format, before executing the DCM framework, we assume an interface extractor to extract the attribute information in the interfaces. For instance, the attribute about title in Figure 1(c) can be extracted as ⟨name = "title of book", domain = any⟩, where "domain = any" means any value is possible. (In this article, we will also address the impact of errors made by the automatic interface extractor on our matching algorithm.) Given extracted raw schema data, we first preprocess schemas to make them ready for mining as the data preprocessing step (Section 3.3). Next, the matching discovery step, the core of the DCM framework, explores a *dual correlation mining* algorithm, which first mines potential attribute groups as positive correlations and then potential complex matchings as negative correlations (Section 3.1). Finally, matching construction ranks and then selects the most confident and consistent matchings from the mining result (Section 3.2). Meanwhile, in the heart of correlation mining, we need to choose an appropriate correlation measure (Section 4).

Further, to complete an automatic matching process, which starts from raw HTML pages as Figure 2 shows, we integrate the DCM framework with an automatic interface extractor [Zhang et al. 2004]. Such "system integration" turns out to be non-trivial– As automatic interface extraction cannot be perfect, it will introduce "noises" (*i.e.*, erroneous extraction), which challenges the performance of the subsequent matching algorithm. As Section 5 will discuss, the errors in the interface extraction step may affect the correlations of matchings and consequently the matching result. In particular, as we will show in the experiments (Section 6.2), the existence of noises may affect the matching accuracy up to 30%.

To make the DCM framework robust against noises, we integrate it with an *ensemble* scheme, which aggregates a multitude of the DCM matchers to achieve robustness, by exploiting statistical sampling and majority voting. Specifically, we randomly sample a subset of schemas (as a *trial*) to match, instead of using all the schemas. Intuitively, it is likely that such a trial still contains sufficient attribute information to match while removing certain noisy schemas. Further, we conduct multiple independent trials. Since errors in different trials are independent, when noises are relatively few, it is likely that only a minority of trials are affected. We thus take majority voting among the discovered matching of all trials to achieve the robustness of holistic matching.

By employing the DCM framework with the ensemble approach, we thus complete an automatic process of matching Web query interfaces. To evaluate the performance of our algorithms, we design two suites of experiments: First, to isolate and evaluate the effectiveness of the DCM framework, we test it on manually extracted TEL-8 dataset in the UIUC Web integration repository [Chang et al. 2003]. The matching result (Section 6.1) shows that the DCM framework achieves good accuracy with perfectly extracted schemas as input. Second, we test the ensemble DCM framework over automatically extracted interfaces in two domains, Books and Airfares, of the TEL-8 dataset. The result shows that the ensemble approach can significantly boost the matching accuracy under noisy schema input, and thus maintain the desired robustness of DCM (Section 6.2).

In summary, the contributions of this article are:

- We build a conceptually **novel connection** between the schema matching problem and the correlation mining approach. On the 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 a **correlation measure** that is particularly suitable for negative correlation. In particular, we identify the problems of existing measures on evaluating negative correlation, which has special importance in schema matching, and further introduce a new correlation measure, H-measure.

- We employ an **ensemble scheme** to connect the DCM framework with automatic interface extractor and thus fully automate the process of matching Web query interfaces. In particular, we identify the need for robust matching and integrate the DCM framework with an ensemble approach by exploiting sampling and voting techniques.

The rest of the article is organized as follows: Section 2 discusses some potential applications of our matching work and further abstracts the problem of matching Web query interfaces. Section 3 develops the base DCM framework. Section 4 proposes a new correlation measure. Section 5 motivates the need for robust matching and discusses the ensemble DCM framework. Section 6 reports our experiments. Section 7 reviews related work and Section 8 concludes the article.

2. CONTEXT AND ABSTRACTION

We observe that our matching scenario, *i.e.*, discovering matchings across "alternative" sources in the same domain (*e.g.*, Books, Airfares), is useful with broad applications. In this section, we first discuss two interesting applications, *MetaQuerier* and *Form Assistant*, in Section 2.1 and then abstract the schema matching problem we are tackling throughout this article in Section 2.2.

2.1 Context

MetaQuerier: We may build a MetaQuerier [Chang et al. 2005] to integrate dynamically discovered and selected alternative sources according to user's queries (http://metaquerier.-cs.uiuc.edu). As [Chang et al. 2005] describes, the goal of the MetaQuerier project is to build a middleware system to help users find and query large scale deep Web sources. Two critical parts of the MetaQuerier system needs the help of the result of matching query interfaces. First, the MetaQuerier needs to build a unified interface for each domain, through which users can issue queries. As also pointed out in [Chang et al. 2005], our work is the basis for building the unified interfaces, *e.g.*, by selecting the most representative attribute among all the attributes that are synonyms of each other. Second, the matchings can be used for improving the quality of source selection according to user's search keywords by expanding the keywords, *e.g.*, author, with semantically related words in the matching result, *e.g.*, writer. Note that in both usage cases, it is not necessary that the matchings be 100% accurate. Although the higher accuracy the result is, the better quality of service we can provide, matching errors can be tolerated in the MetaQuerier system.

Form Assistant: As another scenario, instead of developing a complete MetaQuerier system, we may build a Form Assistant toolkit [Zhang et al. 2005] to help users translate queries from one interface to other relevant interfaces. For instance, if a user fills the query form in *amazon.com*, the Form Assistant can *suggest* translated queries for another interested source such as *bn.com*. To enable such query translation, the Form Assistant needs to first find matching attributes between two interfaces. The matching algorithm in [Zhang et al. 2005] employs a *domain thesaurus* (in addition to simple syntactic similarity-based matching) that specifies the correspondences of attributes in the domain. As mentioned in [Zhang et al. 2005], our matching work can be exploited as an automatic way to construct the domain-specific thesauruses. Note that in this Form Assistant scenario, matching and translation errors can be tolerated, since the objective is to reduce users' effort by providing the "best-effort" query suggestion. Users can correct the errors, if any, before sending the query.

2.2 Abstraction

These potential application scenarios, as the context of our work, indicate a new abstraction for schema matching, *i.e.*, discovering semantic correspondences among a set, instead of a pair, of schemas. As Section 7 will elaborate, existing automatic schema matching works mostly focus on matchings between two schemas (*e.g.*, [Madhavan et al. 2001; Doan et al. 2001]). Based on this fact, the latest survey [Rahm and Bernstein 2001] abstracts schema

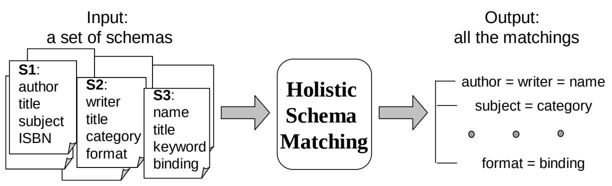


Fig. 3. The holistic schema matching abstraction.

matching as pairwise similarity mappings between two input sources. In contrast, in large scale integration scenarios such as the MetaQuerier and the Form Assistant systems, the schema matching task needs to match many schemas of a domain at the same time and find all matchings at once. We name this new matching setting *holistic schema matching*. As Figure 3 shows, holistic schema matching takes a set of schemas as input and outputs all the matchings among the input schemas. Such a holistic view enables us to explore the *context* information beyond two schemas (*e.g.*, similar attributes across multiple schemas; co-occurrence patterns among attributes), which is not available when schemas are matched only in pairs.

Next, we give an abstraction of the sub-tasks of the matching problem. In general, schema matching can be decomposed into two sequential steps: *Matching discovery* and *matching construction*. To begin with, matching discovery is to find a set of matching candidates that are likely to be correct matchings by exploiting some matching algorithms. This step is often the main focus of schema matching works. Further, matching construction will select the most confident and consistent subset of candidates as the final matching result. A simple score-based ranking of matchings and constraint-based selection strategy is often used in this construction step. Separating our matching problem into the two steps helps to "modularize" our framework, by defining a clean interface between these modules, so that alternative matching discovery (*i.e.*, not necessarily correlation mining-based) or construction techniques can also be incorporated. We will more formally abstract these two steps below.

Matching Discovery

We view a query interface as a flat schema with a set of *attribute entities*. An attribute entity can be identified by attribute *name*, *type* and *domain* (*i.e.*, instance values). Before matching, the data preprocessing step (Section 3.3) finds syntactically similar entities among schemas. Each attribute entity is assigned a unique *attribute identifier*, which we will simply refer to as attribute below. While the matching is over the attribute entities, for our illustration, we use the attribute name of each entity as the attribute identifier. For instance, the schema in Figure 1(c) is thus a set of two attribute entities, written as $\{$title, author$\}$.

Formally, we abstract the matching discovery step as: *Given the input as a set of schemas $I = \{Q_1, ..., Q_N\}$ in the same domain, where each schema Q_i is a set of attributes, find all the matching candidates $R = \{M_1, ..., M_V\}$. Each candidate 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 (i.e., a group of attributes that are semantically equivalent to another attribute or another group of attributes) and $G_{j_k} \subseteq \bigcup_{t=1}^{N} Q_i$.* Semantically, each M_j represents the synonym relationship of attribute groups G_{j_1},..., G_{j_w} and each G_{j_k} represents the grouping relationship of attributes in G_{j_k}.

Sharing the same abstraction of holistic schema matching, different realizations for the matching discovery step have been proposed. In this article, we present a dual correlation mining algorithm for finding matchings as a specific realization. Other approaches include statistical model discovery based approaches [He and Chang 2003], and clustering-based approaches [He et al. 2003; Wu et al. 2004], as Section 7 will discuss.

Matching Construction

Given a set of discovered matching candidates $R = \{M_1, ..., M_V\}$, the matching construction step proceeds with two sub-steps: matching *ranking* and *selection*. First, the matching ranking phase sorts all the candidates with some ranking criterion C. Let us denote the ranked matchings as $R_C = \{M_{t_1}, ..., M_{t_V}\}$. In most schema matching works for finding pairwise semantic correspondences of two schemas (*e.g.*, [Doan et al. 2001; Madhavan et al. 2001]), the matching ranking phase is quite straightforward by directly using the scores computed in the matching discovery step to rank candidates. However, in our scenario of finding n-ary complex matchings, as Section 3.2 will discuss, we find such a simple ranking strategy may not work well and thus develop a different scoring strategy from the one used for finding matching candidates.

Then, the matching selection phase selects a subset of candidates from R_C as the final output of the matching process. Most schema matching works, including ours (Section 3.2), select candidates based on some consistency constraints, *e.g.*, two matching candidates that conflict by covering the same attribute cannot co-exist in the selection. Reference [Melnik et al. 2002] develops a candidate selection strategy based on not only such constraints but also some selection metrics, inspired by bipartite-graph matching.

We have discussed the abstraction of the matching discovery and matching construction steps in our holistic schema matching problem. We will elaborate our realizations of these two steps in the DCM framework, together with the data preprocessing step, in the next section.

3. FROM MATCHING TO MINING: THE BASE *DCM* FRAMEWORK

In this section, we will present technical details of the DCM framework, based on the abstraction Section 2 described. (Section 5 will motivate and discuss a further enhancement, deploying the DCM framework with an ensemble scheme.) In particular, we first focus on the core steps– matching discovery and matching construction– in Section 3.1 and Section 3.2 respectively, and then discuss the data preprocessing step in Section 3.3.

3.1 Matching Discovery: Dual Correlation Mining

We first discuss our development for the core of DCM, *i.e.*, the matching discovery step, as Section 2 identified. We develop a dual correlation mining algorithm, which discovers candidates of complex matchings, *i.e.*, $R = \{M_1, ..., M_V\}$, as mining positive and then negative correlations among attributes across schemas. In particular, the dual correlation mining algorithm consists of *group discovery* and *matching discovery*. 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 cor-*

related 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.

After group discovery, we need to add the discovered groups into the input schemas 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.

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:

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

where m is some correlation measure for two items. That is, we define C_{min} as the minimal value of the pairwise evaluation, thus requiring all pairs to meet this minimal "strength." In principle, any correlation measure for two items is applicable as m (*e.g.*, the measures surveyed in [Tan et al. 2002]); However, since the semantic correctness of the mining result is of special importance for our schema matching task, we develop a new measure, H-measure, which can better satisfy the quality requirements of measures (Section 4).

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.

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*, 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*, if $C_{min}(\{G_1, ..., G_m\}, m_n)$

```
Algorithm: APRIORICORRMINING:
Input: Input Schemas $I = \{Q_1, ..., Q_N\}$, Measure $m$, Threshold $T$
Output: Correlated items
begin:
1  $X \leftarrow \emptyset$
2  /* First, find all correlated two items */
3  /* $\mathcal{V}$: the vocabulary of all items */
4  $\mathcal{V} \leftarrow \cup_{t=1}^{N} Q_i, Q_i \in I$
5  for all $A_p, A_q \in \mathcal{V}, p \neq q$
6      if $m(A_p, A_q) \geq T$ then $X \leftarrow X \cup \{\{A_p, A_q\}\}$
7  /* Then, build correlated $l+1$ items from correlated $l$ items */
8  /* $X_l$: correlated $l$ items */
9  $l \leftarrow 2$
10 $X_l \leftarrow X$
11 while $X_l \neq \emptyset$
12     $X_{l+1} \leftarrow \emptyset$
13     for each item group $Y \in X_l$
14         for each item $A \in \mathcal{V} - Y$
15             /* $Z$: a candidate of correlated $l+1$ items */
16             $Z \leftarrow Y \cup \{A\}$
17             /* Verify whether $Z$ is in fact a correlated $l+1$ items */
18             $Z_l \leftarrow$ all subsets of $Z$ with size $l$
19             if $Z_l \subset X_l$ then $X_{l+1} \leftarrow X_{l+1} \cup Z$
20     $X \leftarrow X \cup X_{l+1}$
21     $X_l \leftarrow X_{l+1}$
21     $l \leftarrow l+1$
22 return $X$
end
```

Fig. 4. Apriori algorithm for mining correlated items.

$\geq T_n$, where m_n is a measure for negative correlation and T_n is another given threshold. As we will discuss in Section 4, although in principle any correlation measure can be used as m_p and m_n, *e.g.*, *Lift* and *Jaccard*, to make matching result more accurate, we need to develop a new correlation measure, which satisfies some crucial characteristics.

Leveraging the apriori feature of C_{min}, we develop Algorithm APRIORICORRMINING (Figure 4) for discovering correlated items, in the spirit of the classic Apriori algorithm for association mining [Agrawal et al. 1993]. Specifically, as the apriori feature indicates, if a set of $l+1$ items $\mathcal{A} = \{A_1, ..., A_{l+1}\}$ satisfies $C_{min}(\mathcal{A}, m) \geq T$, then any subset of $\mathcal{A}$ with size more than one item, denoted as $\mathcal{A}^*$, also satisfies $C_{min}(\mathcal{A}^*, m) \geq T$. It can be shown the the reverse argument is also true. Further, it can be shown that as long as any subset of $\mathcal{A}$ with size l, also denoted as $\mathcal{A}^*$, satisfies $C_{min}(\mathcal{A}^*, m) \geq T$, we have $C_{min}(\mathcal{A}, m) \geq T$ [Agrawal et al. 1993].

Therefore, we can find all the correlated items with size $l+1$ based on the ones with size l. As Figure 4 shows, to start, we first find all correlated two items (Lines 4-6). Next, we repeatedly construct correlated $l+1$ items X_{l+1} from correlated l items X_l(Lines 9-22). In particular, for each correlated l items Y in X_l, we expand Y to be a candidate of correlated $l+1$ items, denoted as Z, by adding a new attribute A (Line 16). We then test whether any

```
Algorithm: DUALCORRELATIONMINING:
Input: Input Schemas $I = \{Q_1, ..., Q_N\}$, 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$(I, m_p, T_p)$
3   /* adding groups into $I$ */
4   for each $Q_i \in I$
5       for each $G_k \in \mathcal{G}$
6           if $Q_i \cap G_k \neq \emptyset$ then $Q_i \leftarrow Q_i \cup \{G_k\}$
7   /* matching discovery */
8   $R \leftarrow$ APRIORICORRMINING$(I, m_n, T_n)$
9   return $R$
end
```

Fig. 5. Algorithm for mining complex matchings.

subset of Z with size l is found in X_l. If so, Z is a correlated $l + 1$ items and added into X_{l+1} (Lines 18-19).

Algorithm DUALCORRELATIONMINING shows the pseudo code of the complex matching discovery (Figure 5). Line 2 (group discovery) calls APRIORICORRMINING to mine positively correlated attributes. Lines 3-6 add the discovered groups into I. Line 8 (matching discovery) calls APRIORICORRMINING to mine negatively correlated attribute groups. Similar to [Agrawal et al. 1993], the time complexity of DUALCORRELATIONMINING 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 Matching Construction: Majority-based Ranking and Constraint-based Selection

After the matching discovery step, we need to develop ranking and selection strategies for the matching construction step, as Section 2 described. We notice that the matching discovery step can discover true semantic matchings and, as expected, also false ones due to the existence of coincidental correlations. For instance, in the Books domain, the result of correlation mining 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 should 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 such 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 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.

Therefore, as Section 2 abstracted, the matching construction step consists of two phases: 1) *Matching ranking*: We need to develop a strategy to reasonably rank to discovered matchings in the dual correlation mining. 2) *Matching selection*: We need to develop a selection algorithm to select the ranked matchings. We will elaborate these two phases in this subsection respectively.

Matching Ranking

To begin with, we need to develop a strategy for *ranking* the discovered matchings R. That is, for any n-ary complex matching M_j in R: $G_{j_1} = G_{j_2} = ... = G_{j_w}$, we have a scoring function $s(M_j, m_n)$ to evaluate M_j under measure m_n and output a ranked list of matchings R_C.

While Section 3.1 discussed a measure for “qualifying” candidates, we now need to develop another “ranking” measure as the scoring 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 scoring function s. Formally,

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

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 scoring 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.

Matching Selection

Given matchings $R_C = \{M_1, ..., M_V\}$ ranked according to the above ranking strategy, we next develop a greedy matching selection algorithm as follows:

1. Among the remaining matchings in R_C, choose the highest ranked matching M_t.

2. Remove matchings conflicting with M_t in R_C.
3. If R_C is empty, stop; otherwise, go to step 1.

Specifically, we select matchings with multiple iterations. In each iteration, we greedily choose the matching with the highest rank and remove its conflicting matchings. The process stops until no matching candidate is left. Example 1 illustrates this greedy selection algorithm with a concrete example. The time complexity of the entire matching construction process is $O(V^2)$, where V is the number of matchings in R_C.

Example 1: Assume running DUALCORRELATIONMINING in the Books domain finds matchings R_C as (matchings are followed by their C_{max} 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.

According to our ranking strategy, we will rank these matchings as $M_1 > M_2 > M_3 > M_4 > M_5 > M_6 > M_7$ in descending order. In particular, although $s(M_1, m_n) = s(M_2, m_n)$, M_1 is ranked higher since $M_1 \succeq M_2$. Next, we select matchings with the greedy strategy: In the first iteration, M_1 is selected since it is ranked the highest. Then we need to remove matchings that conflict with M_1. For instance, M_2 conflicts with M_1 on author and thus should be removed from R_C. 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 selected. M_6 and M_7 are removed because they conflict with M_3. Now R_C is empty and the algorithm stops. The final output is thus M_1 and M_3. ■

3.3 Data Preprocessing

As input of the DCM framework, we assume an interface extractor (Figure 2) has extracted attribute information from Web interfaces in HTML formats. For instance, the attribute about title in Figure 1(c) and Figure 1(d) can be extracted as ⟨name = "title of book", domain = any⟩ and ⟨name = "search for a specific title", domain = any⟩ respectively, where "domain = any" means any value is possible. Section 5 will discuss the incorporation of an automatic interface extractor [Zhang et al. 2004].

As we can see from the above examples, the extracted raw schemas contain many syntactic variations around the "core" concept (*e.g.*, title) and thus are not readily minable. We thus perform a data preprocessing step to make schemas ready for mining. The data preprocessing step consists of *attribute normalization*, *type recognition* and *syntactic merging*. To begin with, given extracted schema data, we perform some standard normalization on the extracted names and domain values. We first stem attribute names and domain values using the standard Porter stemming algorithm [Porter]. 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 3.3.1 discusses, type information helps to identify homonyms (*i.e.*, two attributes may have the same name

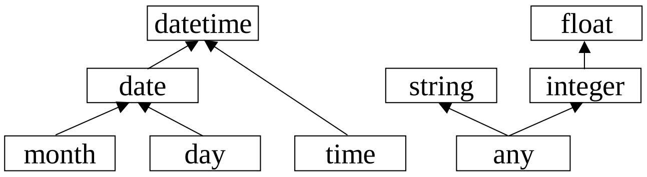


Fig. 6. The compatibility of types.

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 3.3.2 discusses our merging strategy.

3.3.1 *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.

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 6 lists the compatibility of all the types in our implementation.

3.3.2 *Syntactic Merging.* We clean the schemas by merging syntactically similar attribute entities, a common data cleaning technique to identify unique entities [Chaudhuri et al. 2003]. 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 to be *name-similar* to attribute A_q if A_p's name $\supseteq A_q$'s

	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_{++}

Fig. 7. Contingency table for test of correlation.

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 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. For query interfaces, we consider the domain of an attribute as the values that the attribute can select from. Such values are often presented in a selection list or a set of radio buttons in a Web form. 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 the 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 $sim(A_p, A_q) = \frac{\forall w\ in\ both\ V_{A_p}\ and\ V_{A_q}, V_{A_p}(w)+V_{A_q}(w)}{\forall w\ in\ either\ V_{A_p}\ or\ V_{A_q}, V_{A_p}(w)+V_{A_q}(w)}$.

Example 2: Assume there are 3 schemas, S_1 to S_3, in the Airfares domain. S_1 contains attribute entity $\langle$name = “trip type”, type = “string”, domain = $\{$“round trip”, “one way”$\}\rangle$, S_2 contains $\langle$name = “trip type”, type = “string”, domain = $\{$“round trip”, “one way”, “multi-city”$\}\rangle$ and S_3 contains $\langle$name = “ticket type”, type = “string”, domain = $\{$“round trip”, “one way”$\}\rangle$.

The attribute entity trip type (string) occurs in S_1 and S_2 and thus its aggregate values $V_{\text{trip type}}$ = $\{$round: 2, trip: 2, one: 2, way: 2, multi: 1, city: 1$\}$, where each word is followed by frequency. In particular, $V_{\text{trip type}}(round) = 2$ since the word “round” uses in both S_1 and S_2. Similarly, $V_{\text{ticket type}}$ = $\{$round: 1, trip: 1, one: 1, way: 1$\}$. Then, according to our similarity function, we have sim(trip type, ticket type) = $\frac{3+3+3+3}{3+3+3+3+1+1}$ =0.86. ■

4. CORRELATION MEASURE

Since we are pursuing a mining approach, we need to choose an appropriate correlation measure. In particular, to complete our development of the DCM framework, we now discuss the positive measure m_p and the negative measure m_n, which are used as the components of C_{min} (Equation 1) for positive and negative correlation mining respectively (Section 3.1).

As discussed in [Tan et al. 2002], 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* [Brunk 1965] of A_p and A_q contains the number of occurrences of each situation, as Figure 7 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

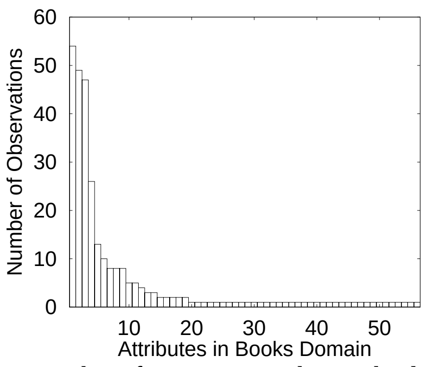


Fig. 8. Attribute frequencies in the Books domain.

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 [Goodman and Kruskal 1979; Tan et al. 2002]. 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. In Section 4.1, we develop a new correlation measure, H-measure, which is better than existing measures in satisfying our requirements. In Section 4.2, we further show that H-measure satisfies a nice sampling invariance property, which makes the setting of parameter threshold independent of the data size.

4.1 H-Measure

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 8 shows the attribute frequencies in the Books domain: First, the non-frequent attributes results in the sparseness of the schema data (*e.g.*, there are over 50 attributes in the Books domain, but each schema only has 5 in average). Second, many attributes are 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 [Tan et al. 2002; Omiecinski 2003; Lee

	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.

Fig. 9. Examples of the three problems.

et al. 2003]. In [Tan et al. 2002], 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 3: Figure 9(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 9(a) shows that *Lift* may give a higher value to less positively correlated attributes. In the scenario of schema matching, the table in Figure 9(a2) should be more positively correlated than the one in Figure 9(a1) because in Figure 9(a2), the co-presence (f_{11}) is more frequently observed than the cross-presence (either f_{10} or f_{01}), while in Figure 9(a1), the co-presence has the same number of observations as the cross-presence. However, *Lift* cannot reflect such preference. In particular, Figure 9(a1) gets a much higher *Lift* and Figure 9(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 [Tan et al. 2002] and the χ^2 measure in [Brin et al. 1997]. 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 4 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 4: Figure 9(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 9(b) shows that *Jaccard* may not be able to distinguish the situations of rare attribute. In particular, Jaccard considers the situations in Figure 9(b1) and Figure 9(b2) as the same. But Figure 9(b2) is more negatively correlated than Figure 9(b1) because A_p in Figure 9(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.

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

A special case in practice is that when one of f_{11}, f_{10}, f_{01} is 0. When f_{11} is 0, H value is always 1; when f_{10} or f_{01} is 0, H values is always 0. Consequently, H value cannot distinguish the subtle differences of correlations under such situations. To avoid this problem, like the solution taken in many other correlation measures, we simply assign f_{11}, f_{10}, f_{01} value 1 if they are 0.

H-measure satisfies the quality requirements: On the 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.

Example 5: To see how H-measure can avoid the sparseness problem and rare attribute problem, let us apply H-measure for the situations in Figure 9(a) and (b). First, for the table in Figure 9(a1), we have H = 0.25 and for the table in Figure 9(a2), we have H = 0.07. Therefore, H-measure considers the table in Figure 9(a2) as more positively correlated, which is consistent with what we want. Second, for the table in Figure 9(b1), we have H = 0.49 and for the table in Figure 9(b2), we have H = 0.92. Therefore, H-measure considers the table in Figure 9(b2) as more negatively correlated, which is also correct. ■

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 the 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 6: Figure 9(c) illustrates the frequent attribute problem with an example, where we still use *Jaccard* as the measure. Figure 9(c) shows that *Jaccard* may give a higher value to independent attributes. In Figure 9(c1), A_p and A_q are independent and both of them have the probabilities 0.9 to be observed; while, in Figure 9(c2), A_p and A_q are really positively correlated. However, *Jaccard* considers Figure 9(c1) as more positively correlated than Figure 9(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.

4.2 Sampling Invariance: Threshold Independence of Data Size

In our dual correlation mining algorithm, we have several thresholds to set for m_p and m_n. Since data integration is fuzzy in nature, like most other schema matching works, such threshold setting seems inevitable. However, the H-measure (and several other correlation measures) has a nice property that enables a wider applicability of threshold settings, by their independence of data size. Specifically, we notice that H value stays unchanged when f_{11}, f_{10}, f_{01} and f_{00} increase or decrease proportionally at the same rate, which indicates that H value is independent of the data size. We name this independence of data size *sampling invariance* property[3]. More formally, we say a correlation measure CM has the sampling invariance property if CM satisfies the following equality:

$$CM(nf_{11}, nf_{10}, nf_{01}, nf_{00}) = CM(f_{11}, f_{10}, f_{01}, f_{00}). \quad (5)$$

Intuitively, the sampling invariance property suggests that the size of input data does not affect the value of the correlation measure as long as the data are "uniformly" scaled down or up. Therefore, given a large set of schema data, if we randomly sample a subset of the schemas and measure the H value over the subset, the value should be the same as that of all the schemas. However, as most measures require "sufficient observations," the sampling size should not be too small. Otherwise, the sampled data may not be sufficiently representative to the original data (in a statistical sense).

This sampling invariance feature alleviates us much effort on parameter tuning for testing our DCM approach. As an automatic schema matching algorithm to cope with a large set of sources, our approach should be able to avoid tuning parameters (*i.e.*, T_m, T_p and T_d) for different sets of input sources. We find the sampling invariance feature of the H-measure can achieve such a goal, since the size of input does not affect the correlation value as long as it is sufficient to represent "the reality." (Our experiment shows that 20 to 30 sources are often quite sufficient to be a representative set, which is not difficult to collect.) Therefore, we can obtain the optimal parameters by tuning them for one dataset

[3]Since H-measure satisfies the sampling invariance property, m_n and m_p, as the two measures we used in the dual correlation mining algorithm, also satisfy the sampling invariance property.

and then these settings can be reusable for any sampled subset of sources (as the ensemble DCM framework in Section 5 will exploit).

Further, we find that an optimal setting obtained in one domain is also reusable for other domains. Our survey for deep Web sources [Chang et al. 2004] shows that different domains often have similar occurrence behavior of attributes. In particular, all the 8 surveyed domains have very similar trends of vocabulary growth over sources and attribute frequency distribution. Therefore, it is possible to tune the parameters in one domain and use them for other domains. In our experiment (Section 6), we tune the parameters in Books and then reuse the setting for all the other 7 domains. The result shows that such a "cross-domain" setting can indeed achieve good performance.

Finally, we notice that besides H-measure, several other measures, *e.g.*, *Lift* and *Jaccard*, also satisfy the sampling invariance property, while other measures such as χ^2 do not satisfy this property. Therefore, the sampling invariance property can also be a criterion for choosing an appropriate measure for correlation mining.

5. DEALING WITH NOISES: THE ENSEMBLE *DCM* FRAMEWORK

To fully automate the matching process, which starts from raw HTML pages as Figure 2 shows, we must integrate the DCM framework (Section 3) with an automatic interface extractor. It turns out that such integration is not trivial– As automatic interface extractor *cannot* be perfect, it will introduce "noises," which challenges the performance of the subsequent DCM matching algorithm. This section presents a refined algorithm, the *ensemble* DCM framework, in contrast to the *base* DCM framework in Section 3, to *maintain* the robustness of DCM against such noisy input.

We note that such "system integration" issues have not been investigated in earlier works. Most works on matching query interfaces, for instance our earlier work [He and Chang 2003] and others [He et al. 2003; Wu et al. 2004], all adopt manually extracted schema data for experiments. While these works rightfully focus on isolated study of the matching module to gain specific insight, for our goal of constructing a fully automatic matching process, we must now address the robustness problem in integrating the interface extraction step and the matching algorithm.

In particular, we integrate our DCM algorithm with the interface extractor we developed recently [Zhang et al. 2004], which tackles the problem of interface extraction with a parsing paradigm. The interface extractor as reported in [Zhang et al. 2004] can typically achieve 85-90% accuracy– thus it will make about 1-1.5 mistake for every 10 query conditions to extract. While the result is quite promising, the 10-15% errors (or *noises*) may still affect the matching quality. As our experiment shows in Section 6.2, with noisy schemas as input, the accuracy of the base DCM framework may degrade up to 30%.

The performance degradation results mainly from two aspects: First, noises may affect the qualification of some correlations and decrease their C_{min} values (*i.e.*, Equation 1) below the given threshold. In this case, the dual correlation mining algorithm cannot discover those matchings. Second, noises may affect the right ranking of matchings (with respect to the C_{max} measure, *i.e.*, Equation 2) and consequently the result of greedy matching selection. Although in principle both qualification and ranking can be affected, the influence on qualification is not as significant as on ranking. Matching qualification will be affected when there are enough noisy schemas, which make the C_{min} value lower than the given thresholds T_p or T_n. In many cases when only a few noises exist, the affected matchings are still above the threshold and thus can be discovered in the qualification step. However, the ranking of matchings using C_{max} is more subtle– That is, even when there are only few noises, the ranking of matchings is still likely to be affected (*i.e.*, incorrect matchings maybe ranked higher than correct ones). The reason is that other than comparing matchings to a fixed threshold, the ranking step needs to compare matching among themselves. A single noise is often enough to change the ranking of two conflict matchings. Consequently, the ranking is less reliable for the matching selection step to choose correct matchings. As a result, although correct matchings may be discovered by the dual correlation mining process, they may be pruned out by the matching selection phase due to the incorrect ranking of matchings, and thus the final matching accuracy degrades.

While large scale schema matching brings forward the inherent problem of noisy quality in interface extraction, the large scale also lends itself to an intriguing potential solution. An interesting question to ask is: *Do we need all input schemas in matching their attributes*? In principle, since pursuing a correlation mining approach, our matching techniques exploit "statistics-based" evaluation in nature and thus need only "sufficient observations." As query interfaces tend to share attributes, *e.g.*, author, title, subject, ISBN are repeatedly used in many book sources, a subset of schemas may still contain sufficient information to "represent" the complete set of schemas. Thus, the DCM matcher in fact needs only sufficient correct schemas to execute, instead of all of them. This insight is promising, but it also brings a new challenge: As there is no way to differentiate noisy schemas from correct ones, how should we select input schemas to guarantee the robustness of our solution?

Tackling this challenge, we propose to extend DCM in an *ensemble* scheme we developed recently [He and Chang 2005], with sampling and voting techniques. (Figure 10 shows this extension from base DCM framework, *i.e.*, Figure 10(a), to ensemble DCM framework, *i.e.*, Figure 10(b), which we will elaborate in Section 5.1.) To begin with, we consider to execute the DCM matcher on a randomly sampled subset of input schemas. Such a *downsampling* has two attractive characteristics: First, when schemas are abundant, the downsampling is likely to still contain sufficient correct schemas to be matched. Second, by sampling away some schemas, it is likely to contain fewer noises and thus is more probable to sustain the DCM matcher. (Our analysis in Section 5.1 attempts to build analytic understanding of these "likelihoods.")

Further, since a single downsampling may (or may not) achieve good result, as a randomized scheme, its expected robustness can only be realized in a "statistical" sense– Thus, we propose to take an ensemble of DCM matchers, where each matcher is executed over an independent downsampling of schemas. We expect that the majority of those ensemble matchers on randomized subsets of schemas will perform more reliably than a single matcher on the entire set. Thus, by taking majority voting among these matchers, we can achieve a robust matching accuracy.

We note that, our ensemble idea is inspired by *bagging classifiers* [Breiman 1996] in machine learning. Bagging is a method for maintaining the robustness of "unstable" classification algorithms where small changes in the training set result in large changes in prediction. In particular, it creates multiple versions of a classifier, trains each classifier on a random redistribution of the training set and finally takes a plurality voting among all the classifiers to predict the class. Therefore, our ensemble approach has the same foundation as bagging classifiers on exploiting majority voting to make an algorithm robust against

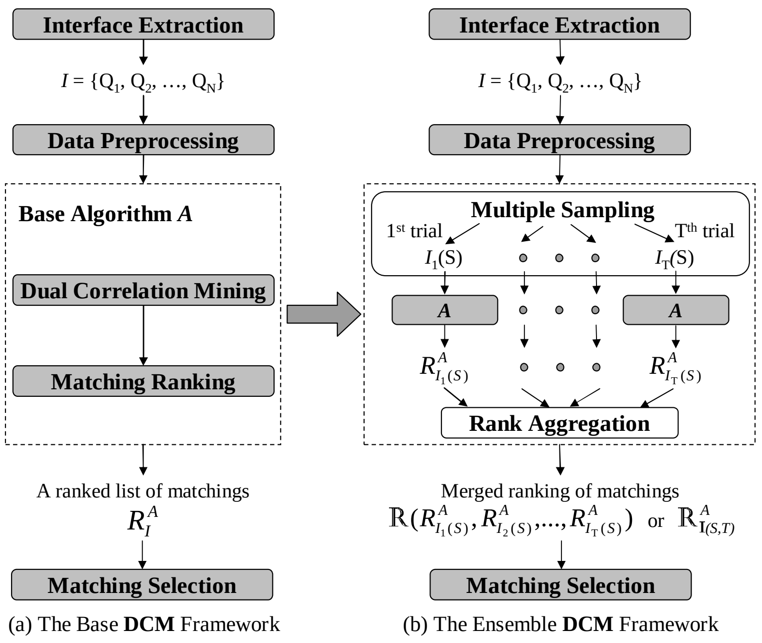

Interface Extraction
$I = \{Q_1, Q_2, \ldots, Q_N\}$
Data Preprocessing
Base Algorithm *A*
Dual Correlation Mining
Matching Ranking
A ranked list of matchings
R_I^A
Matching Selection
(a) The Base **DCM** Framework
Multiple Sampling
1st trial
Tth trial
$I_1(S)$
$I_T(S)$
A
$R^A_{I_1(S)}$
$R^A_{I_T(S)}$
Rank Aggregation
Merged ranking of matchings
$\mathbb{R}(R^A_{I_1(S)}, R^A_{I_2(S)}, \ldots, R^A_{I_T(S)})$ or $\mathbb{R}^A_{\mathbf{I}(S,T)}$
(b) The Ensemble **DCM** Framework

Fig. 10. From the base DCM framework to the ensemble DCM framework.

outlier data in the input.

However, our approach is different from bagging classifiers in several aspects. First, *setting*: We apply the idea of the ensemble of randomized data for unsupervised learning (*e.g.*, in our scenario, schema matching with statistical analysis), instead of supervised learning (*i.e.*, human experts give the learner direct feedback about the correctness of the performance [Langley 1995]), which bagging classifiers is developed for. Second, *techniques*: Our concrete techniques are different from bagging classifiers. In particular, in the sampling part, we take a downsampling other than random redistribution with replacement; in the voting part, we need to aggregate a set of ranked lists, which is more complicated than aggregate a set of labels in bagging classifiers. Third, *analytic modeling*: We build an analytic modeling specific to our matching scenario (Section 5.1), which enables us to validate the effectiveness of a particular configuration and thus can be the basis for the design of the ensemble scheme.

We will next discuss this ensemble DCM framework in details. In particular, we first more formally model this framework and analyze its effectiveness (Section 5.1). Then, we aggregate the results of multiple DCM matchers with a voting algorithm, which thus essentially captures the consensus of the majority (Section 5.3).

5.1 Analytical Modeling

We develop a general modeling to formalize the ensemble DCM framework just motivated. Our goals are two fold: First, based on our modeling, we can analytically judge the effectiveness of the ensemble approach. Second, the modeling can be used to validate the setting of parameters in the ensemble scheme.

We first redraw the base DCM framework in Figure 2 as Figure 10(a) by expanding the two steps in matching construction, *i.e.*, matching ranking and matching selection. We view the dual correlation mining algorithm DUALCORRELATIONMINING and the matching ranking together as a black box *base algorithm* A. As we just discussed, the performance degradation is mainly caused by the impact of noises on A, where the output of A, denoted by R_I^A (*i.e.*, the output ranking determined by A over input I), is disturbed. The goal of our ensemble DCM framework is thus to make A still output reasonably good ranking of matchings with the presence of noises.

Specifically, given a set of N schemas I as input, assume there are W problematic schemas (*i.e.*, noises) that affect the ranking of M. Suppose the holistic matcher A can correctly rank M if one trial draws no more than K noises ($K < W$)– *i.e.*, in which case, M as a correct matching can actually be ranked higher.

Next, we need to model the ensemble framework, which consists of two steps: *multiple sampling* and *rank aggregation*, as Figure 10(b) shows. *First*, in the multiple sampling step, we conduct T downsamplings of the input schemas I, where each downsampling is a subset of independently sampled S schemas from I. We name such a downsampling as a *trial* and thus have T trials in total. We denote ith trial as $I_i(S)$ $(1 \leq i \leq T)$. By executing the base algorithm A over each trial $I_i(S)$, we get a ranked list of matchings $R^A_{I_i(S)}$. *Second*, the rank aggregation step aggregates ranked matchings from all the trials, *i.e.*, $R^A_{I_i(S)}$ $(1 \leq i \leq T)$, into a merged list of ranked matchings, which we denote as $\mathbb{R}(R^A_{I_1(S)}, ..., R^A_{I_T(S)})$, or $\mathbb{R}^A_{\mathbb{I}(S,T)}$ in short. We expect the aggregate ranking $\mathbb{R}^A_{\mathbb{I}(S,T)}$ can alleviate the impact of noises and thus is better than R_I^A.

Since W is determined by "inherent" characteristics of input schemas I and K by the holistic matcher A, we name them as *base parameters*. Unlike W and K, the sampling size S and the number of trials T are "engineered" configurations of the ensemble framework and thus named as *framework parameters*.

Our goal of analysis is thus to justify, given estimation of the base parameters, W and K, which characterize the data quality and the base algorithm, can certain configuration, in terms of S and T, of the ensemble framework achieve robustness? (If so, we will then ask, how to determine appropriate settings of S and T, which is the topic of Section 5.2.)

In particular, given our modeling, we can derive the probability to correctly rank M in a single trial, which we name as *hit probability*, *i.e.*, the chance of "hit" a correct ranking of M in a single trial (and as we will discuss later, we will do more trials to enhance the overall hit ratio). Given base parameters W and K of M, hit probability is a function of S (and not T as it is for a single trial) and thus denoted as $\alpha_M(S)$. To derive $\alpha_M(S)$, we first compute the probability that there are exactly i noises in a single trial, denoted by $Pr(k = i|S)$, *i.e.*, with i noises out of W and $S - i$ correct ones out of $N - W$:

$$Pr(k = i|S) = \frac{\binom{W}{i}\binom{N-W}{S-i}}{\binom{N}{S}} \tag{6}$$

As our model assumes, M can be correctly ranked when there are no more than K noises. We thus have:

$$\alpha_M(S) = \sum_{i=0}^{K} Pr(k = i|S) \tag{7}$$

Next, we are interested in how many times, among T trials, can we observe M being ranked correctly? (This derivation will help us to address the "reverse" question in Section 5.2: To observe M in a majority of trials with a high confidence, how many trials are necessary?) This problem can be transformed as the standard scenario of tossing an unfair coin in statistics: Given the probability of getting a "head" in each toss as $\alpha_M(S)$, with T tosses, how many times can we observe heads? With this equivalent view, we know that the number of trials in which M is correctly ranked (*i.e.*, the number of tosses to observe heads), denoted by O_M, is a random variable that has a binomial distribution [Anderson et al. 1984] with the success probability in one trial as $\alpha_M(S)$. We use $Pr(O_M = t|S, T)$ to denote the probability that M is correctly ranked in exactly t trials. According to the binomial distribution, we have

$$Pr(O_M = t|S, T) = \frac{T!}{t!(T-t)!}\alpha_M(S)^t(1-\alpha_M(S))^{T-t} \tag{8}$$

Since our goal is to take majority voting among all the trials (in rank aggregation), we need a sufficient number of trials to ensure that M is "very likely" to be correctly ranked in the relative majority of trials. As an analogy, consider the coin tossing: Even the probability to get a head in each toss is high, say 0.8, we may not always observe $0.8 \times T$ heads in T trials; the actual number of heads may even be a minority of trials– And our goal is to design a T such that "the number of heads" is very likely to be the majority. We thus need a sufficient number of trials to enable the majority voting. We name the probability that M can be correctly ranked in the majority of trials (*i.e.*, more than half of trials) as *voting confidence*. Voting confidence is a function of T (as just intuitively observed) and S (as it also depends on $\alpha_M(S)$ and thus S). We denote the voting confidence as $\beta_M(S, T)$. In particular, we have

$$\beta_M(S, T) = \sum_{t=\frac{T+1}{2}}^{T} Pr(O_M = t|S, T). \tag{9}$$

As a remark, in Equation 9, we constrain T as an odd number and thus $\frac{T+1}{2}$ is the minimum number of trials needed to be the majority[4].

Our modeling essentially captures the functional relationship of the sampling size S and the number of trials T to together achieve a desired voting confidence. There are two interpretations of Equation 9 in examining a framework: First, given S and T, we can use Equation 9 to evaluate how effective the framework is. In particular, we illustrate with Example 7 as a basis of understanding how the framework works. Second, we can use Equation 9 to design the configuration of a framework. That is, for an objective voting confidence to achieve, what would be the right configuration of S and T? Section 5.2 will focus on this configuration issue.

[4]When T is odd, the notion of majority is always well defined, as there are no ties (of equal halves). This advantage ensures there is no ambiguous situation in comparing two matchings in the rank aggregation step in Section 5.3. Also, when T is odd, $\beta_M(S, T)$ becomes a monotonic function of T. We use this property to derive an appropriate configuration in Section 5.2.

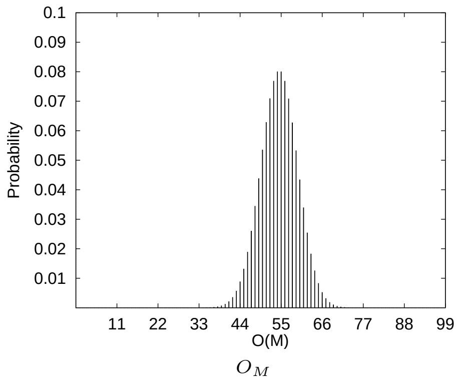


Fig. 11. The binomial distribution of O_M, with $T = 99$ and $\alpha_M(S) = 0.55$.

Example 7: Assume there are 50 input schemas (*i.e.*, $N = 50$). As characteristics of the data quality and the base algorithm, suppose a matching M cannot be correctly ranked because of 6 noisy schemas (*i.e.*, $W = 6$); on the other hand, suppose M can be correctly ranked if there are no more than two noisy schemas (*i.e.*, K = 2). Also, as the configuration of the ensemble framework, suppose we want to sample 20 schemas in a single trial and conduct 99 trials (*i.e.*, $S = 20$ and $T = 99$).

According to Equation 6, in any single trial, we have 0.04 probability to get no noisy schema, 0.18 probability with one and 0.33 probability with two. Together, we have 0.04 + 0.18 + 0.33 = 0.55 probability to correctly rank M in one trial (*i.e.*, $\alpha_M(S) = 0.55$).

Further, Figure 11 shows the binomial distribution of O_M. Going back to the coin tossing analogy, this figure essentially shows, if the probability to get a head in one toss is 0.55, after tossing 99 times, the probability of observing a certain number of heads. For instance, we have $Pr(O_M = 50|S, T) = 0.05$, which means the probability to observe 50 heads in 99 tosses is 0.05. According to Equation 9, we have 0.84 voting confidence to correctly rank M (or observe heads) in more than 49 trials (or tosses) (*i.e.*, $\beta_M(S, T)$ = 0.84). Therefore, even $\alpha_M(S)$ is not very high, *e.g.*, 0.55 in this example, with sufficient number of trails, it is still very likely that M can be correctly ranked in the majority of trials. ■

Finally, while our analysis above focuses on a single matching, there are multiple matchings, M_1, M_2, ..., M_n, to discover. We note that our analysis can generally assume a representative "worst-case" matching, based on which the analysis will also cover all the matchings. Specifically, the above modeling can be applied to any M_i with its corresponding W_i and K_i values. We then assume there is a "worst-case" matching M^* with base parameters W^* and K^*. We want to show that if we are likely to correctly rank M^* in the majority of trials under some setting, we are even more likely to correctly rank all the matchings M_1, M_2, ..., M_n in the majority of trials with the same setting. If this statement can be justified, we only need to consider the "worst-case" situation in determining the ensemble configuration in Section 5.2.

We show that the base parameters of the imaginary "worst-case" matching M^* can be set as $W^* = \max W_i$ and $K^* = \min K_i$, $1 \leq i \leq n$. Intuitively, the higher W is, the lower $\alpha_M(S)$ will be because we have more noises in the input schemas I; on the other hand, the lower K is, the lower $\alpha_M(S)$ will be because the base algorithm A is less robust against noises. More formally, we can show that $\alpha_M(S)$ is monotonically decreasing with respect to W and monotonically increasing with respect to K. (The derivation is straightforward and thus we do not provide a proof here.) Therefore, if we assume a matching M^* with base parameters W^* as the maximal value of W_i and K^* the minimal value of K_i, we have $\alpha_{_{M_i}}(S) \geq \alpha_{_{M^*}}(S)$ any matching M_i ($1 \leq i \leq n$).

Further, we can show that all the matchings also have higher voting confidence than M^*. Intuitively, if a matching M has higher hit probability, M should be more likely to be observed in the majority of trials, which means it also has a higher voting confidence. In particular, we can show that $\beta_{_M}(S, T)$ is monotonically increasing with respect to $\alpha_{_M}(S)$. (Similarly, the derivation is straightforward and thus we do not provide a proof here.) Therefore, since $\alpha_{_{M_i}}(S) \geq \alpha_{_{M^*}}(S)$ ($1 \leq i \leq n$), we have $\beta_{_{M_i}}(S, T) \geq \beta_{_{M^*}}(S, T)$ ($1 \leq i \leq n$). This inequality indicates that M^* is indeed the "worst-case" matching. Specifically, if we can find an appropriate setting of S and T to correctly rank M^* in the majority of trials with high confidence, we will have even more confidence to correctly rank all the matchings in the majority of trials with the same setting of S and T.

5.2 Sampling and Trials: Configuration

This section focuses on the first phase of the ensemble framework: Sampling and trials. The key challenge we need to address is: Given W and K, we need to find an appropriate configuration of S and T to provide guarantee on voting confidence. To begin with, we must characterize our "system environment" by estimating the base parameters W and K. Then, we discuss our strategy to configure S and T based on our modeling in Section 5.1.

Base Parameters: Before deriving S and T, we need to estimate the "worst-case" base parameters W^* and K^* in Equations 6 and 7. In particular, W^* and K^* can be related to the error rate and the tolerance threshold respectively in the modeling of error cascade. First, as W^* characterizes the noisy degree of the input schemas I, we let $W^* = N \times \rho$, where N is the number of schemas and ρ is the error rate of I. In our development, we set ρ to 0.1, as the worst-case value, according to the accuracy of current interface extraction technique, as discussed earlier. Second, since the behavior of A is very specific and complicated, it may be difficult to accurately obtain K^*. We thus take a conservative setting, which will lead to a "safe" framework, *e.g.*, setting the worst-case K^* as a small constant.

As just discussed, all matchings that are not worse than the worst-case setting can be guaranteed to have higher voting confidences. Therefore, with conservative worst-case settings, we expect to correctly rank more matchings in the aggregate result $\mathbb{R}^A_{\mathbb{I}(S,T)}$.

Framework Parameters: In Section 5.1, we have shown that, for some matching M with respect to given base parameters W and K, for certain framework parameters S and T, we can derive the voting confidence $\beta_{_{M^*}}(S, T)$ with statistical analysis. Now we are facing the reverse problem: Given estimated W, K, and our objective voting confidence, what are the appropriate S and T values we should take? Formally, *given W, K, and an objective voting confidence c, what are the sampling size S and the number of trials T we should take to ensure M^* has at least a probability of c to be correctly ranked in the majority of trials, i.e., $\beta_{_{M^*}}(S, T) \geq c$?*

In particular, we want to know, among all the (S, T) pairs that satisfy the above statement, which pair is the most appropriate? To answer this question, we need to develop some criteria to evaluate settings. Intuitively, we would like to prefer a (S, T) pair that can maximize S and minimize T:

On the one hand, we want to reduce unnecessary downsampling. A very small S value may not be able to collect enough schemas to represent the complete input data and consequently degrade the accuracy of the base algorithm A. It may also, by overly-aggressive downsampling, remove some more "unique" (but correct) schemas from consideration, and thus reduce the applicability of the matching result. Thus, among all the valid (S, T) pairs, we prefer a larger S that can cover more schemas.

On the other hand, we want to reduce unnecessary trials. As Section 5.1 discussed, the more trials we have, the higher voting confidence will be. We can formally show that when T is limited to be odd, $\beta_{_{M^*}}(S, T)$ is monotonically increasing with respect to T. (Again, the derivation is straightforward and thus we do not provide a proof here.) Considering the execution time of the ensemble framework, we do not want to be over-tried; therefore, among all the valid (S, T) pairs, we prefer a pair with a smaller T.

However, these two goals cannot be optimized at the same time, because as our modeling shows, S and T are not independent– One will negatively affect the choice of another. Specifically, when we set $\beta_{_{M^*}}(S, T)$ to an objective confidence c, T can be viewed as a function of S or vice versa. Choosing one will thus also affect another: A larger S will result in a lower hit probability and thus more trials T for the same objective confidence; on the other hand, a smaller T will demand a higher hit probability and thus a smaller sampling size S. Consequently, in the space of all valid (S, T) pairs, there does not exist one that can optimize both S and T.

To balance these two goals, we have to choose a trade-off setting. We propose two ways to determine S and T:

First, $S \rightarrow T$: In some situations, we may have a reasonable grasp of S, so that we know the range of input size (*i.e.*, the degree of downsampling) that the base algorithm may demand– *e.g.*, some statistical approach typically requires a minimal number of observations of data to ensure its statistical confidence. In such a case, we can start with an S value and set T as the minimal (odd) number that can achieve the objective confidence c, *i.e.*,

$$T = \min\{t | t > 0, t \text{ is odd}, \beta_{_{M^*}}(S, T)(S, t) \geq c\} \tag{10}$$

Second, $T \rightarrow S$: In other situations, we may be constrained by affordable computation time, which determines the acceptable range of T. In this case, we start with a desired number of trials T and choose the maximal S to achieve the objective confidence, *i.e.*,

$$S = \max\{s | 1 \leq s \leq N, \beta_{_{M^*}}(S, T)(s, T) \geq c\} \tag{11}$$

Example 8: Assume there are 50 input schemas (*i.e.*, $N = 50$) and our objective confidence is 0.9 (*i.e.*, $c = 0.9$). According to our discussion, the settings of the "worst-case" matching M^* are W^* = $N \times \rho$ = $50 \times 0.1 = 5$ and $K^* = 2$. Setting K^* to 2 is a "safe" configuration we also use in our experiments (Section 6).

In the $S \rightarrow T$ strategy, assume we set S = 20. Based on our modeling, for any odd number t, we can compute the voting confidence $\beta_{_{M^*}}(S, t)$. According to Equation 10, we take the minimal t that satisfies $\beta_{_{M^*}}(S, t) \geq 0.9$ and thus we get $T = 11$.

On the other hand, in the $T \rightarrow S$ strategy, assume we set T = 41. Similarly, for any s ($1 \leq s \leq N$), we can compute $\beta_{_{M^*}}(s, T)$. According to Equation 11, we take the maximal

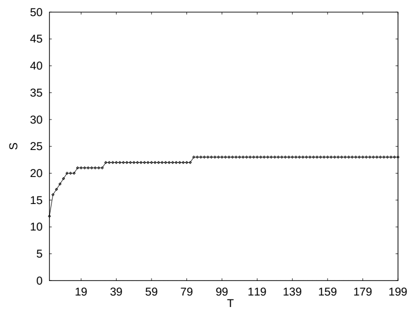


Fig. 12. The insensitivity of S on T.

S that satisfies $\beta_{M^*}(s, T) \geq 0.9$ and thus we get $S = 22$. ∎

Although both $S \rightarrow T$ and $T \rightarrow S$ are valid configuration strategies, as Example 8 just showed, in practice the $T \rightarrow S$ strategy is better because it is easier to pick T. To illustrate this statement, we compute the corresponding S values for all the odd numbers T between 0 to 200 using the $T \rightarrow S$ strategy, *i.e.*, Equation 11, with the same system setting as Example 8 assumed. Figure 12 shows the result, from where we observe that when T increase to some point, around 30 in this example, the corresponding S values become very stable, almost insensitive to the change of T.

On the other hand, from the same Figure 12, we can infer the opposite trend of the $S \rightarrow T$ strategy. Picking an S will significantly affect the value of T. Some S values may result in a very large T, which is not affordable in practice. In some cases, for a large S, maybe it is even impossible to find a corresponding T that satisfies the given objective voting confidence.

Overall, it is much easier to pick T than S in practice. Therefore, in our experiments (Section 6), we adopt the $T \rightarrow S$ strategy. Also, we will show that the empirical result of testing the framework with various configuration settings is consistent with our analysis above.

5.3 Voting: Rank Aggregation

This section discusses the second phase of the ensemble framework: Aggregating rankings $R^A_{I_1(S)}$, ..., $R^A_{I_T(S)}$ from the T trials into a merged list of ranked matchings $\mathbb{R}^A_{\mathbb{I}(S,T)}$. The main issue we are facing is to develop a rank aggregation strategy that can reflect the majority "consensus" in $\mathbb{R}^A_{\mathbb{I}(S,T)}$.

We notice that this rank aggregation in our situation is slightly different from the traditional rank aggregation problem. Traditional rank aggregation assumes all voters share the same set of candidates and only rank them in different orders. In contrast, in our scenario, no candidate is given before executing the base algorithm and each trial outputs its own matching result. Therefore, before aggregate rankings, we need to have a candidate selection step to select matching candidates.

Consequently, the rank aggregation phase consists of two sub-steps: 1) Candidate selection: To select candidates from each $R^A_{I_i(S)}$ to form a common pool of candidates $\mathcal{C}$. 2) Rank aggregation: To aggregate the T rankings $PR^A_{I_1(S)}$, ..., $PR^A_{I_T(S)}$ into $\mathbb{R}^A_{\mathbb{I}(S,T)}$, where $PR^A_{I_i(S)}$ is the "projected" ranking of $R^A_{I_i(S)}$ on $\mathcal{C}$, as we will discuss below.

Candidate Selection

We select candidates based on the intuition that if a matching M is only discovered by a minority of trials, M is more likely to be a false matching. Therefore, we consider a matching as a candidate if it appears in the majority of T rankings, $R^A_{I_1(S)}$, ..., $R^A_{I_T(S)}$. All the matchings whose number of occurrences are less than $\frac{T+1}{2}$ are thus pruned.

Let $\mathcal{C}$ denote the union of all the candidates in each $R^A_{I_i(S)}$. After candidate selection, we will remove the non-candidate matchings from each $R^A_{I_i(S)}$ and meanwhile preserving the ordering of candidates; the corresponding new ranked list, which can be viewed as a "projection" of $R^A_{I_i(S)}$ on $\mathcal{C}$, contains only candidates and is denoted as $PR^A_{I_i(S)}$.

Example 9: Assume we execute the base algorithm A on three trials, *i.e.*, $T = 3$, and the outputs are thus three ranked lists $R^A_{I_1(S)}$, $R^A_{I_2(S)}$ and $R^A_{I_3(S)}$. Suppose $R^A_{I_1(S)}$ outputs ranking $M_1 > M_2 > M_3 > M_4$ in descending order, $R^A_{I_2(S)}$ outputs $M_2 > M_1 > M_3 > M_5$, and $R^A_{I_3(S)}$ outputs $M_3 > M_1 > M_2 > M_4$.

Since $\frac{T+1}{2} = 2$, any matching that occurs only once will be pruned. In particular, M_5 is pruned; other matchings, M_1, M_2, M_3 and M_4, all at least occur twice and thus are selected as matching candidates. Therefore, we have $\mathcal{C} = \{M_1, M_2, M_3, M_4\}$.

The projected rankings are thus $RR^A_{I_1(S)}$: $M_1 > M_2 > M_3 > M_4$, $PR^A_{I_2(S)}$: $M_2 > M_1 > M_3$, and $PR^A_{I_3(S)}$: $M_3 > M_1 > M_2 > M_4$. In particular, M_5 does not appear in $PR^A_{I_2(S)}$ because it has been pruned. ■

Rank Aggregation

In rank aggregation, we need to construct an ordered list $\mathbb{R}^A_{\mathbb{I}(S,T)}$ for the candidates in $\mathcal{C}$, based on the individual ranks $PR^A_{I_1(S)}$, ..., $PR^A_{I_T(S)}$. This problem is essentially a *rank aggregation* problem, which has been extensively studied as a particular *voting* system in both social science [Kemeny 1959; Young 1988] and computer science [Dwork et al. 2001; Fagin et al. 2003]. In the literature, many rank aggregation strategies have been proposed, such as Borda's aggregation [Borda 1781], Kemeny optimal aggregation [Kemeny 1959], and footrule optimal aggregation [Dwork et al. 2001]. There does not exist an aggregation strategy that can beat other strategies in all aspects– Different strategies have different strength and weakness.

Before discussing concrete aggregation strategies, we first need to solve the partial list problem. Specifically, since the output of one trial may not contain all the candidates in $\mathcal{C}$, $PR^A_{I_i(S)}$ may be only a partially ordered list. To be able to apply the aggregation strategy (as we will discuss below), it is necessary to also assign ranks to the candidates not in the list. In our development, given a trial with a partial list, we assign all the uncovered candidates with the same lowest rank. Therefore, in one trial, a covered candidate is always ranked higher than an uncovered one, and two uncovered candidates are equally ranked.

Since essentially any rank aggregation strategy can be applied in our scenario, in our development, we test several different aggregation strategies and our goal is to find the most appropriate one. We first choose the widely deployed Borda's aggregation [Borda 1781] as the baseline aggregation strategy. We then realize that to enforce the majority voting, it is important that an aggregation strategy satisfies the *Condorcet criterion* [Young 1988]. We thus propose a strategy, *FK aggregation*, by combining Kemeny optimal aggregation [Kemeny 1959] and footrule optimal aggregation [Dwork et al. 2001]. We will discuss these two strategies, *i.e.*, Borda's aggregation and FK aggregation, in details respectively.

5.3.1 *Baseline: Borda Aggregation.* A primary strength of Borda's aggregation is that it is rather simple and computationally efficient: It can be implemented in linear time. Borda's aggregation also satisfies some good properties such as anonymity, neutrality, and consistency [Young 1974]. Specifically, in Borda's aggregation, given a candidate M_j, let r_{ji} be the number of matchings ranked lower than M_j in $PR^A_{I_i(S)}$, the *borda score* of M_j, denoted as $B(M_j)$, is defined as the sum of all r_{ji}, *i.e.*, $B(M_j) = \sum_{k=1}^{T} r_{jk}$. The aggregation result $\mathbb{R}^A_{\mathbb{I}(S,T)}$ is thus the descending ordering of all the candidates with respect to their borda scores.

Example 10: Continue on Example 9, after candidate selection, we first complete the partial lists. In particular, since $PR^A_{I_2(S)}$ only partially ranks the four candidates, we assign the lowest rank to the uncovered candidate M_4, *i.e.*, we rank M_4 as the $4th$ candidate in $PR^A_{I_2(S)}$. Next, we compute the borda score for each candidate and then apply Borda's aggregation. In particular, since M_1 is ranked higher than 3 candidates in $PR^A_{I_1(S)}$, 2 in $PR^A_{I_2(S)}$ and 2 in $PR^A_{I_3(S)}$, the borda score for M_1 is 3 + 2 + 2 = 7. Similarly, the borda scores for M_2 to M_4 are 6, 5, 0 respectively. The final ranking $\mathbb{R}^A_{\mathbb{I}(S,T)}$ is thus $M_1 > M_2 > M_3 > M_4$. ■

5.3.2 *Enforcing Majority by Satisfying the Condorcet Criterion: FK Aggregation.* Our analysis of the effectiveness of the ensemble DCM framework in Section 5.1 is based on the assumption that when a matching is correctly ranked in the majority of trials, it will be correctly ranked in $\mathbb{R}^A_{\mathbb{I}(S,T)}$. Therefore, our aggregation strategy should reflect this requirement of majority– That is, if a matching can be correctly ranked in most trials, its ranking in $\mathbb{R}^A_{\mathbb{I}(S,T)}$ should also be correct.

We notice that this requirement is consistent with the classic *Condorcet criterion* [Young 1988]. Specifically, the Condorcet criterion requires that, given any two candidates M_i and M_j, if a majority of trials ranks M_i higher than M_j, then M_i should be ranked higher than M_j in the aggregate list $\mathbb{R}^A_{\mathbb{I}(S,T)}$. (As we can see here, setting the number of trials T as an odd number, as Section 5.1 discussed, can ensure that there will be no tie situation between any two M_i and M_j.) The fact that aggregation mechanisms that satisfy the Condorcet criterion can yield robust results has also been noticed and exploited by [Dwork et al. 2001]. However, Borda's aggregation, although computationally very fast, does not satisfy the Condorcet criterion. To our knowledge, the only aggregation strategy exactly satisfies the Condorcet criterion is Kemeny optimal aggregation. Another strategy, footrule optimal aggregation, does not directly satisfy the Condorcet criterion, but its ordering of matchings yields a factor-2 approximation to Kemeny optimal aggregation.

Example 11: To see how Borda's aggregation may not satisfy the Condorcet criterion, let us see an example, which is slightly different from Example 9. Assume after candidate selection, we have $RR^A_{I_1(S)}$: $M_1 > M_2 > M_3 > M_4$, $PR^A_{I_2(S)}$: $M_1 > M_2 > M_4 > M_3$, and $PR^A_{I_3(S)}$: $M_2 > M_3 > M_4$.

With Borda's aggregation, we have the borda scores for M_1, M_2, M_3 and M_4 as 6, 7, 3, 2 respectively. The ranking of matchings under Borda's aggregation is thus $M_2 > M_1 > M_3 > M_4$. However, M_1 is ranked higher than M_2 in the majority of trials, *i.e.*, $RR^A_{I_1(S)}$ and $RR^A_{I_2(S)}$, which shows that Borda's aggregation violates the Condorcet criterion and therefore may not reflect the results of majority. ■

Although Kemeny optimal aggregation satisfies the Condorcet criterion, it is computationally expensive. Kemeny optimal aggregation is to find the ordered list $\mathbb{R}^A_{\mathbb{I}(S,T)}$ that minimizes $\sum_{i=1}^{T} K(PR^A_{I_i(S)}, \mathbb{R}^A_{\mathbb{I}(S,T)})$, where $K(PR^A_{I_i(S)}, \mathbb{R}^A_{\mathbb{I}(S,T)})$ denotes the *Kendall tau* distance. That is, it is the number of pairs of candidates (M_i, M_j) on which the ordered lists $PR^A_{I_i(S)}$ and $\mathbb{R}^A_{\mathbb{I}(S,T)}$ disagree (*i.e.*, one ranks M_i higher than M_j, while another one ranks M_j higher than M_i). It has been proven that computing Kemeny optimal aggregation is NP-Hard [Dwork et al. 2001], which is not affordable in practice. Hence, we cannot only apply this aggregation strategy.

As the approximation to Kemeny optimal aggregation, footrule optimal aggregation has good computational complexity. In footrule optimal aggregation, the aggregate list $\mathbb{R}^A_{\mathbb{I}(S,T)}$ contains the median ranks of all the matchings. Specifically, given a candidate M_j, let q_{ji} be the rank of M_j in $PR^A_{I_i(S)}$, the *median rank* of M_i is defined as $medain(M_j) = median(q_{j1}, ..., q_{jT})$. The aggregation result $\mathbb{R}^A_{\mathbb{I}(S,T)}$ is thus the ordered list of median ranks of all the candidates. Footrule optimal aggregation can be computed in polynomial time. Although it may not satisfy the Condorcet criterion, it has been shown that its ordering of matchings (*i.e.*, the footrule distance) has a factor-2 approximation to the Kendall tau distance in Kemeny optimal aggregation [Diaconis and Graham 1977]. However, footrule optimal aggregation suffers the tie problem. That is, some matchings may have the same median rank and it is unclear how to break ties in footrule optimal aggregation.

Combining the strength of these two aggregation strategies, in our development, we develop a hybrid aggregation strategy, *FK aggregation*. In particular, we first apply footrule optimal aggregation. To break a tie, we apply Kemeny optimal aggregation only locally for ranking the candidates that cause the tie. Empirically, since the number of candidates result in a tie is often very few (*e.g.*, less than 4), the computation is very efficient.

Example 12: Let us apply FK aggregation for the case in Example 11. We first complete the partial lists. In particular, since $PR^A_{I_3(S)}$ only partially rank the four candidates, we assign the lowest rank to the uncovered candidate M_1.

We then compute the median rank for each candidate and apply footrule optimal aggregation. In particular, the median rank for M_1 is median(1, 1, 4) = 1. Similarly, the median ranks for M_2 to M_4 are 2, 3, 3 respectively.

Since M_3 and M_4 get a tie in footrule optimal aggregation, we break the tie by applying Kemeny optimal aggregation only on M_3 and M_4. Since two out of the three trials prefer M_3 than M_4, we rank M_3 higher than M_4. The final ranking $\mathbb{R}^A_{\mathbb{I}(S,T)}$ is thus $M_1 > M_2 > M_3 > M_4$, which is consistent with the result of only applying Kemeny optimal aggregation, but more efficient. ■

6. EXPERIMENTS

We evaluate the DCM framework over real query interfaces. In particular, we implement all the algorithms in Python 2.4 and test all the experiments on a Windows XP machine with Pentium M 1.6GHz CPU and 512M memory. We choose the TEL-8 dataset of the UIUC Web integration repository [Chang et al. 2003] as the testbed of both the base DCM framework and the ensemble one. The TEL-8 dataset contains raw Web pages for 447 deep Web sources in 8 popular domains, where each domain has about 20-70 sources.

Experiment Suites

To evaluate the performance of the algorithms we have developed in this article, we design two suites of experiments. The first suite of experiments is to evaluate the effectiveness of the base DCM framework by testing it on manually extracted interfaces. Our goal is to isolate the matching process to study and thus fairly evaluate its performance. In particular, we assume a perfect interface extractor to extract all the interfaces in the TEL-8 dataset into query capabilities by manually doing the extraction work. After extracting the raw data, we do the data preprocessing according to the process explained in Section 3.3. Then, we run the dual correlation mining and matching construction algorithms on the preprocessed data in each domain. In the results, we use attribute name and type together as the attribute identifier for an attribute entity since we incorporate the type recognition step to identify homonyms (Section 3.3). The experimental result shows that the base DCM framework achieves good *target accuracy* on manually extracted schemas. We also evaluate the effectiveness of the matching selection algorithm and the data preprocessing step. Further, we compare the H-measure with other measures on the same dataset and the result shows that H-measure outperforms the others in most cases. Section 6.1 will report these results.

The second suite of experiments is to verify the impact of noises in the interface extraction on our matching algorithm and evaluate the performance of the ensemble approach. In particular, we conduct our evaluations on automatically extracted interfaces in two domains: Books and Airfares. First, we directly run the base DCM framework on automatically extracted interfaces as the baseline result that we will compare to. Second, we measure the accuracy of the ensemble DCM framework and compare it to the baseline result. The experiments show that the ensemble approach can significantly improve the matching accuracy of DCM. Third, we execute the ensemble DCM framework under various parameter settings and compare the empirical values with our theoretical analysis. Section 6.2 will report these results.

Note that, to illustrate the effectiveness of the dual correlation mining, we only list and count the "semantically difficult" matchings discovered by the mining algorithm, not the simple matchings by the syntactic merging in the data preprocessing (*e.g.*, merging "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.

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 [He and Chang 2003], by customizing the *target questions* to the complex matching scenario. The idea of the target accuracy is to measure how accurately 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 the 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 13: For instance, for matching $\{A\} = \{B, C\}$, the closenym sets of attributes A,

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	R	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		{author (any)} = {last name (any), first name (any)}		0.87
		{subject (string)} = {category (string)}		0.83

Fig. 13. Running dual correlation mining and matching construction on the Books domain.

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 [He and Chang 2003]. Specifically, we assume a "random querier" to ask for the closenym set of each attribute according to its frequency. The answer to each question is the 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:

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

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

6.1 Base *DCM* Framework

In this subsection, we report the first suite of experiment: The evaluation of the base DCM framework.

Result on the TEL-8 dataset: In this experiment, we run our algorithm (with H-measure as the correlation measure) on manually extracted 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 in the Books domain. As we discussed in Section 4.2, the reason we do not need to consider the number of input schemas in the parameter setting is

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
MusicRecords	title (unknown) = album (unknown)	Y
	keyword (unknown) = {song (unknown), album (unknown)}	N
	genre (string) = category (string)	Y
Hotels	{check in (date), check out (date)} = {check in date (date), check out date (date)}	Y
	check in (date) = check in date (date)	Y
	check out (date) = check out date (date)	Y
	type (string) = country (string)	N
	guest (integer) = {adult (integer), child (integer), night (integer)}	P

Fig. 14. Experimental results for Airfares and Movies.

that H-measure satisfies the sampling invariance property. We then observe that different domains often have similar popularity and occurrence behavior of attributes, as our deep Web survey [Chang et al. 2004] studies. Therefore, we reuse the setting of parameters in the Books domain for all the other domains in both the base DCM and the ensemble DCM frameworks. The sampling invariance property of H-measure and the observation of attribute distribution similarity across domains greatly alleviate our effort on parameter tuning for developing an automatic query interface matcher for various sampling sizes and different domains.

Figure 13 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 13). 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 13). We can see that M_1 and M_3 are fully correct matchings, while others are partially correct or incorrect. Last, the matching construction will choose M_1 and M_3 (*i.e.*, the correct ones) as the final output.

Figure 14 shows the results of some other domains, *i.e.*, Airfares, Movies, Music Records and Hotels. 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 the 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 8). 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* F (*i.e.*, the number of occurrences of the attribute over the total number of schemas is above F) in both discovered matchings and correct matchings to measure the target accuracy. Specifically, we run the algorithms on all the

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

Fig. 15. Target accuracy of the 8 domains.

Domain	reduced false positive (20%)	missed false positive (20%)	reduced false positive (10%)	missed false positive (10%)
Books	0	0	3	0
Airfares	2	0	22	0
Movies	0	0	2	0
MusicRecords	3	0	5	1
Hotels	6	1	11	2
CarRentals	2	1	2	1
Jobs	4	0	9	1
Automobiles	0	0	2	1

Fig. 16. The effectiveness of reducing false matchings in the matching selection step.

attributes and then report the target accuracy in terms of the frequency-divided attributes. In the experiment, we choose F 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 14 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 14 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 15 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.

Evaluating the Matching Selection Strategy: To evaluate the effectiveness of the matching selection algorithm we developed in Section 3.2, which exploits conflict between matching candidates to remove false positives, we count the number of false matchings

Domain	P_T (20%)	R_T (20%)	P_T (10%)	R_T (10%)
Books	*0.79* (**-0.21**)	1	*0.74* (**-0.26**)	1
Airfares	1	1	*0.81* (**-0.19**)	*0.82* (+0.11)
Movies	1	1	*0.87* (**-0.13**)	1
MusicRecords	*0.93* (**-0.07**)	1	*0.70* (**-0.06**)	1
Hotels	*0.66* (**-0.20**)	1	*0.47* (**-0.39**)	*0.46* (**-0.41**)
CarRentals	*1* (+0.28)	*0.63* (**-0.37**)	*1* (+0.28)	*0.16* (**-0.44**)
Jobs	*0.70* (**-0.30**)	*1* (+0.14)	*0.52* (**-0.26**)	0.87
Automobiles	1	1	*0.66* (**-0.27**)	*0.68* (**-0.32**)

Fig. 17. Target accuracy of the 8 domains without data preprocessing.

reduced and missed by the selection step respectively. Figure 16 lists this result for the eight domains under both 20% and 10% frequency thresholds. We can see that the greedy selection strategy based on C_{max} measure is quite effective in reducing false matchings. Most false matchings are removed in the selection step. In particular, although the 10% frequency threshold may result in more false matchings comparing to the 20% one, the selection strategy can remove most of them and keep the performance good. For instance, in the Airfares domain under 10% frequency threshold, 22 false matchings are removed and no false matching is missed.

Evaluating the Data Preprocessing Step: To evaluate the effectiveness of the data preprocessing step, we test the DCM algorithm over schemas without data preprocessing. In particular, we only perform the standard normalization sub-step in Section 3.3 for the input schemas and ignore the type recognition and syntactic merging sub-steps. Our goal is to see the impact of these sub-steps on the accuracy of matching.

Intuitively, although query interfaces are quite concerted in terms of the attributes they use, there still are many syntactic variations for expressing the same attribute, *e.g.*, title, book title, title of book and search by title for attribute "title." As discussed in Section 3.3, type recognition and syntactic merging can help merge these variations into a single attribute and thus increase attribute occurrences across query interfaces, which can improve the performance of the subsequent correlation mining algorithm.

Figure 17 shows the result of running the DCM algorithm with non-preprocessed schemas as input. In Figure 17, we write accuracies that change after removing the data preprocessing step in italic font and show the differences in brackets. As we can see, the accuracies for many domains are much worse than the ones with data preprocessing in Figure 15. In particular, under the 10% frequency threshold, where more attributes are considered for mining matchings, accuracies are greatly reduced. Therefore, applying the data preprocessing step, although may itself result in some errors, is crucial for our mining-based matching approach and can indeed significantly enhance the matching accuracy.

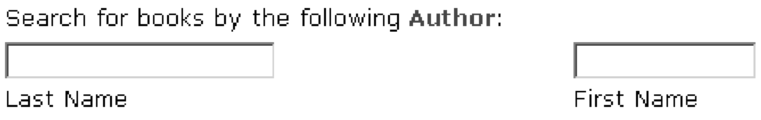
Evaluating the H-measure: We compare the H-measure with other measures and the result shows that the H-measure gets better target accuracy. As an example, we choose *Jaccard* (ζ) as the measure we compare to. With *Jaccard*, we define the m_p and m_n as

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

and

Domain	$P_T(H)$ (10%)	$R_T(H)$ (10%)	$P_T(\zeta)$ (10%)	$R_T(\zeta)$ (10%)
Books	1	1	*0.80* (**-0.20**)	1
Airfares	1	0.71	*0.79* (**-0.21**)	*0.61* (**-0.10**)
Movies	1	1	*0.93* (**-0.07**)	1
MusicRecords	0.76	1	0.76	1
Hotels	0.86	0.87	*0.44* (**-0.42**)	*0.95* (+0.08)
CarRentals	0.72	0.60	*0.68* (**-0.04**)	*0.62* (+0.02)
Jobs	0.78	0.87	*0.64* (**-0.14**)	0.87
Automobiles	0.93	1	*0.78* (**-0.15**)	1

Fig. 18. Comparison of H-measure and *Jaccard*.


Search for books by the following **Author**:

Last Name

First Name

Fig. 19. An example of incorrectly extracted query interfaces.

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

We set the T_p and T_n for this *Jaccard*-based m_p and m_n as 0.5 and 0.9 respectively. (Since *Jaccard* also satisfies the *sampling invariance* property, we can similarly find the optimal thresholds within one domain and the values are reusable for other domains.) We compare the target accuracy of the H-measure and *Jaccard* in the situation of 10% frequency threshold. The result (Figure 18) shows that the 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*.

6.2 Ensemble *DCM* Framework

In this subsection, we report the second suite of experiment: The evaluation of the ensemble DCM framework. All the experiments in this subsection are conducted with the setting of frequency threshold as 20% (*i.e.*, F = 20%).

The baseline matching result: The baseline result we will compare to is executing the base DCM algorithm on automatically extracted interfaces. In particular, we use the techniques in [Zhang et al. 2004] to extract interfaces in two domains, Books and Airfares. The second and third columns in Figure 20 show the result, where the second column is the target precision and the third column the target recall.

We can see that the accuracies of the baseline approach degrades up to 30%, comparing to the results in Figure 15. This performance degradation is mainly because the existence of noises affects the qualification and ranking of matchings and thus the result of matching selection. For instance, in the Books domain, author = last name is ranked higher than author = $\{$last name, first name$\}$ because in some interfaces (*e.g.*, the ones shown in Figure 19), the input box which should be associated with "Last Name" is incorrectly associated with "Search for books by the following Author". Such errors lower down the negative correlation between author and first name and thus result in the selection of the partially correct matching author = last name.

Domain	*The base DCM framework*		*The ensemble DCM framework* with Borda's aggregation				*The ensemble DCM framework* with FK aggregation			
	P_T	R_T	P_{AT}	R_{AT}	P_{FT}	R_{FT}	P_{AT}	R_{AT}	P_{FT}	R_{FT}
Books	0.73	0.75	0.83	0.89	0.9	1.0	0.83	0.9	0.9	1.0
Airfares	0.67	0.68	0.79	0.79	0.71	0.82	0.91	0.73	1.0	0.73

Fig. 20. The comparison of target accuracy on Books and Airfares.

Also, due to the greedy selection strategy, the errors caused in one iteration may cascade to its subsequent iterations. For instance, still in the Books domain, when author = $\{$last name, first name$\}$ is pruned out (because of the above reason), in the next iteration of selection, isbn = $\{$last name, first name$\}$ is selected as a correct matching, which makes the result even worse.

The performance of the ensemble DCM framework: Before running the ensemble framework, we need to first determine its configuration. In our experiment, we choose the $T \rightarrow S$ configuration strategy developed in Section 5.2. Specifically, we set the number of trials T as 41 and objective voting confidence c as 0.9 for both Books and Airfares. (As we modeled in Section 5.1, T is set as an odd number. We have no particular reason for choosing 41. As Section 5.2 discussed, S is insensitive to T and thus picking other T values will not significantly affect the final performance. We also empirically verify this fact later.) We then set W^* and K^* values according to our estimation strategy of the base parameters. In particular, for Books, we have W^* = 6 and for Airfares, W^* = 5. For both domains, we set K^* as a small constant 2. Thus, according to Equation 11, we have S = 22 for Books and S = 19 for Airfares. Also, for each dataset, we test it with the two aggregation strategies we developed in Section 5.3 respectively: The Borda's aggregation and the FK aggregation.

As the ensemble framework is essentially a data-randomized approach (with multiple random trials), it is "non-deterministic"– We thus measure the distribution of its performance. Specifically, we execute the framework 100 times on Books with the same setting S = 22, T = 41. Similarly, we execute it 100 times on Airfares with the same setting S = 19, T = 41. To quantify the comparison with the baseline result, we measure two suites of target accuracies: the *average target accuracy* (*i.e.*, the average precision and recall of the 100 executions, denoted as P_{AT} and R_{AT} respectively) and the *most frequent target accuracy* (*i.e.*, the most frequently obtained precision and recall of the 100 executions, denoted as P_{FT} and R_{FT} respectively). Note that we do not use the best target accuracy (*i.e.*, the best precision and recall of the 100 executions) as we did in [He and Chang 2005] because in practice we cannot judge which result is the best without knowledge from human experts. In contrast, most frequent accuracy is more meaningful since it can be obtained by executing the ensemble framework multiple times and taking their majority.

The results of both average and most frequent accuracies are listed in Figure 20 (columns 3-6 for Borda's aggregation and columns 7-10 for FK aggregation). We can see that: 1) Comparing to the baseline result, precision and recall are improved by the ensemble framework under both aggregation strategies. 2) For the Books domain, Borda's aggregation and FK aggregation have roughly the same accuracy; For the Airfares domain, FK aggregation can achieve much higher precision than Borda's aggregation, but with slightly lower recall.

Overall, the ensemble framework is quite effective in maintaining the robustness of the DCM matcher. The FK aggregation strategy can yield more robust results than Borda's aggregation. We believe this experiment shows that, while Borda is actually a reasonable

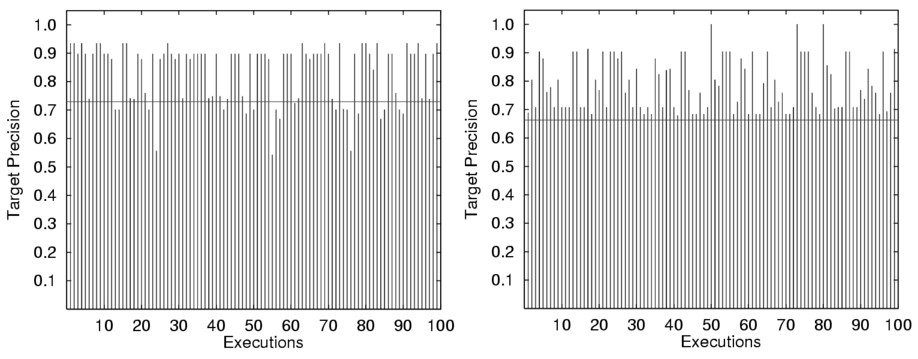

(a) Books. (b) Airfares.

Fig. 21. The target precision with 100 executions on two domains (Borda's aggregation).

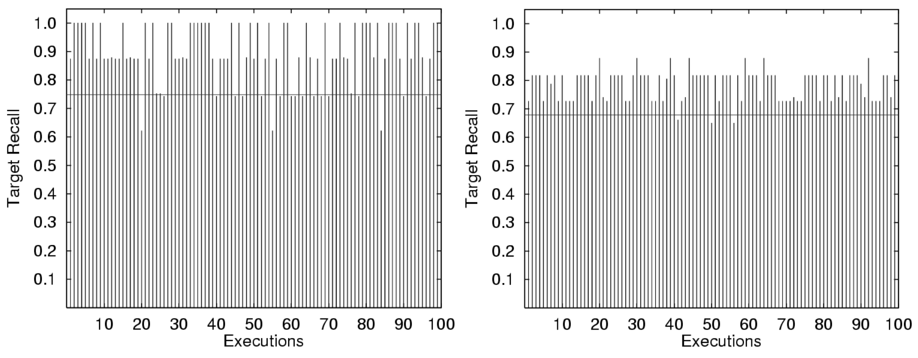

(a) Books. (b) Airfares.

Fig. 22. The target recall with 100 executions on two domains (Borda's aggregation).

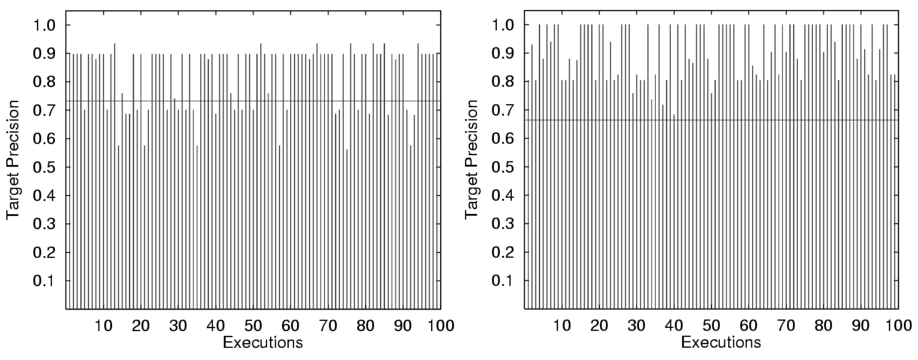
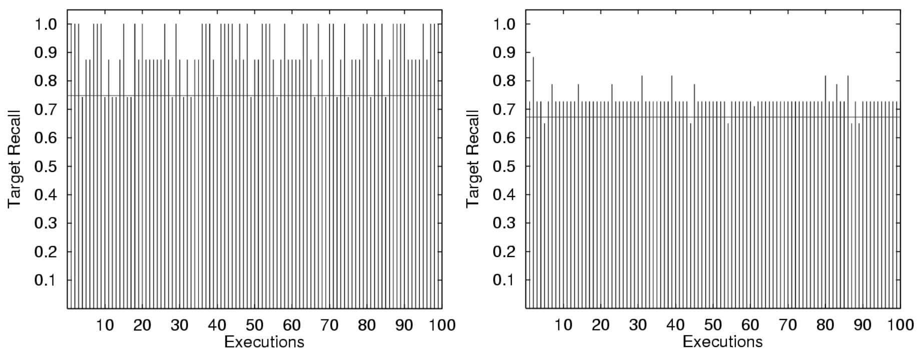

(a) Books. (b) Airfares.

Fig. 23. The target precision with 100 executions on two domains (FK aggregation).

baseline choice, FK is indeed more robust. Next, we illustrate and interpret the results of the ensemble framework with more details.

First, in most executions, the ensemble framework achieves better accuracy than the baseline result. For instance, Figure 21 shows the 100 target precisions of the 100 executions over Books and Airfares with Borda's aggregation. To make Figure 21 more illustrative, we use straight horizontal lines to denote the baseline accuracies. We can see



(a) Books. **(b)** Airfares.

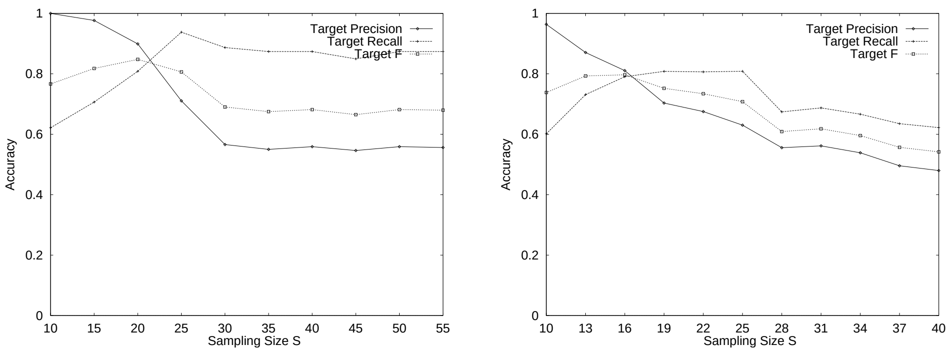
Fig. 24. The target recall with 100 executions on two domains (FK aggregation).

that, although accuracies may be varying in different executions, most precisions in both Books and Airfares are better than their corresponding baseline precisions. Similar result can also be observed in the target recall part (Figure 22) under Borda's aggregation and both precision (Figure 24) and recall (Figure 24) under FK aggregation. Hence, this experiment indicates that the ensemble framework can indeed boost the matching accuracy under noisy schema input, and thus maintain the desired robustness of a holistic matcher. Note that the recall graphs looks more regular than the precision ones because for recall, only the value on numerator is changing, while for precision, values on both numerator and denominator are changing.

Second, from Figures 21 to 24, we also observe an interesting phenomenon: It seems that there are upper-bounds for both precision and recall, which the ensemble framework cannot exceed. The existence of such upper bounds is because, in essence, there are two types of data quality problems, noises and missing data, and the ensemble framework can deal with noises, but not missing data. Specifically, noises are some observed data that ideally should not be observed, *i.e.*, *outliers*. For instance, the extraction of a book schema, *e.g.*, the one in Figure 19, may incorrectly consider "author" as an attribute and thus lowers down the correlation of "author" and "first name." Although noises may affect the accuracy of the base algorithm, they are minority in quantity. Downsampling is thus a good approach to filtering them out and, consequently, the majority voting can be effective. On the other hand, *missing data* are some data that ideally should be observed, but in reality are not. For instance, the input schemas may contain only a small number of occurrences of the attribute "last name" and thus we cannot sufficiently identify to find the grouping of "last name" and "first name." For this missing data case, sampling and voting techniques will not help, since when the entire dataset has missing data, all the trials will also have missing data and their aggregate result cannot fix the problem. The ensemble framework, with the limit imposed by such missing data, has an upper bound for the best accuracy.

Finally, the execution time of the ensemble framework is also acceptable. The 100 executions on Books take 118 seconds for Borda's aggregation and 117 seconds for FK aggregation. The 100 executions on Airfares take 109 seconds for Borda's aggregation and 128 seconds for FK aggregation. Therefore, the average time for one execution is about only 1 second.

The result under various configuration settings: The purpose of this set of experiments



(a) Books (T=41). **(b)** Airfares (T=41).

Fig. 25. The target accuracy under various sampling sizes (Borda's aggregation).

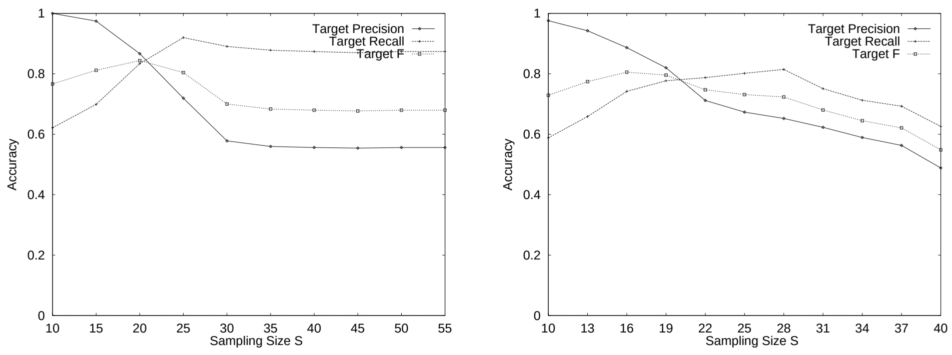
is to empirically verify our analysis in Section 5.2: 1) We want to verify whether our setting of S using Equation 11 is consistent with empirical observation. 2) We want to verify whether the performance of the framework is indeed insensitive to T, but sensitive to S.

First, we measure the accuracy of the ensemble framework with different sampling sizes on the two domains. In particular, we fix T at 41 and let S progressively increase from 10 to 55 with an increment size 5 (*i.e.*, 10, 15, 20, ..., 55) for Books and from 10 to 40 with an increment size 3 for Airfares. For each sampling size, we execute the ensemble framework 30 times under the two aggregation strategies respectively and compute the average precisions and recalls. Figure 25 shows the experimental result under Borda's aggregation and Figure 26 FK aggregation.

From Figures 25 and 26, we can observe the same trend in both domains, which seems to be independent of the aggregation strategy we choose. Specifically, when sampling size increases, the target precision mostly keeps on decreasing, while the target recall goes up first and then goes down at some point. We give the explanation as below: A small sampling size may miss some attributes in downsampling and thus discover less matchings, which results in trivially high precision but low recall. With larger sampling size, we are able to cover more attributes and thus discover not only more correct matchings, but also a few false matchings. Consequently, the precision decreases and recall increases. When the sampling size is too large, a downsampling is likely to have many noises and thus the recall starts to decrease again.

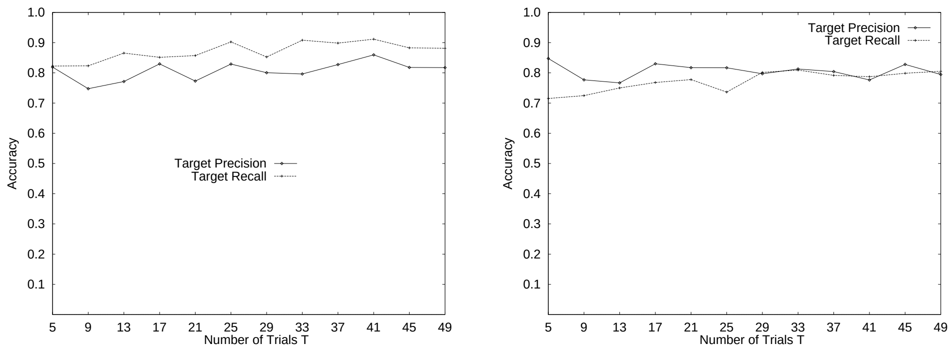
The best sampling size we should take is thus some values in the middle. We choose the F-measure, which combines precision P and recall R as $F = \frac{2PR}{P+R}$, to measure the overall accuracy. From Figure 25(a) and Figure 26(a), we can see the best range of sampling size for Books, according to F-measure, is around 20. Our setting based on Equation 11 is 22, which is quite close to 20. Similarly, from Figure 25(b) and Figure 26(b), the best range of sampling size for Airfares is around 16. Our setting based on Equation 10 is 19, which is also close. Therefore, our configuration strategy of determining the sampling size is consistent with the empirical result.

Second, since we choose T as 41 with no particular reason in the experiment, we want to verify that the choosing other T values is in fact not quite different, because of the insensitivity of S on T (Section 5.2). In particular, we fix S at 22 for Books and 19 for Airfares. We change T from 5 to 49 with increment size 4 for both domains. For each T,



(a) Books (T=41). **(b)** Airfares (T=41).

Fig. 26. The target accuracy under various sampling sizes (FK aggregation).



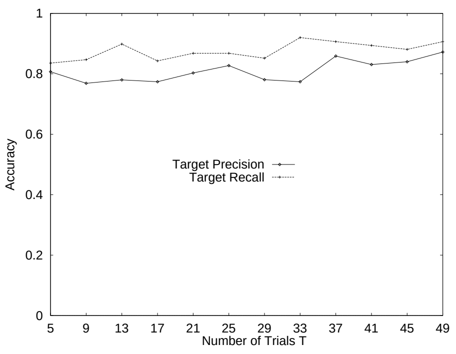
(a) Books (S=22). **(b)** Airfares (S=19).

Fig. 27. The target accuracy under various number of trials (Borda's aggregation).

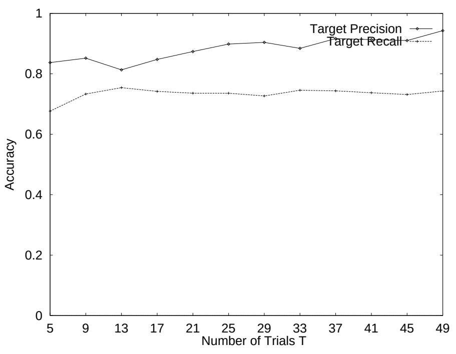
we again execute the framework 30 times under the two aggregation strategies and compute the average precisions and recalls. Figures 27 and 28 shows the experimental results. From the results, we can see that, in both domains, both the precision and recall become more and more flat and stable when T increases. This result indicates that with other T values (as long as it is not too small), we can also have roughly the same performance and thus the decision on T is not a critical factor.

Also, comparing Figure 27 and Figure 28, we can observe that the ensemble framwork with FK aggregation generally can achieve better precision than the one with Borda's aggregation. This result indicates that FK aggregation is more robust than Borda's aggregation in dealing with noisy data, since it approximates the Condorcet criterion (Section 5.3).

Overall, from these two experiments on S and T, we can see that under the same T, different sampling sizes S will significantly affect the performance of the data-ensemble framework, while on the other hand, under the same S, different number of trials T have little impact on the performance. This sensitivity of performance on S but not T indicates that the $T \rightarrow S$ configuration strategy is better than $S \rightarrow T$ because T is much easier to pick in practice, which verifies our analysis in Section 5.2.



(a) Books (S=22).



(b) Airfares (S=19).

Fig. 28. The target accuracy under various number of trials (FK aggregation).

7. RELATED WORK

We relate our work to the literature in two aspects: the *problem* and our *techniques*. In particular, in terms of the problem, we discuss the difference of our work with 1) traditional schema matching works; 2) recent works on matching Web databases. In terms of our techniques, we discuss the distinctions of 1) our mining techniques; 2) our ensemble matching approach.

First, in terms of the *problem*: Schema matching is important for schema integration [Batini et al. 1986; Seligman et al. 2002] and thus has got great attention. However, existing schema matching works mostly focus on simple 1:1 matching [Rahm and Bernstein 2001; Doan et al. 2001; Madhavan et al. 2001] 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. A recent work iMAP [Dhamankar et al. 2004] proposes to construct 1:n matchings between two schemas by combining their simple 1:1 matchings. Although both aiming at complex matchings, our work is different from iMAP in: 1) *scenario*: iMAP focuses on matching two schemas, while we targets at large scale schema matching. 2) *techniques*: iMAP relies on the availability of instance values to construct complex matchings from simple 1:1 matchings, while we explore the co-occurrence information across schemas and thus develop a correlation mining approach.

Some recent works are particularly focusing on matching Web databases [He et al. 2003; Wu et al. 2004; Wang et al. 2004]. WISE [He et al. 2003] is a comprehensive query interface integrator, which evaluates the similarity of attributes in multiple aspects. However, it only deals with simple 1:1 matchings. Reference [Wang et al. 2004] matches query interfaces based on the results of probing some instance values from the back-end databases via interfaces. It also only deals with simple 1:1 matchings. Comparing with other matching approaches, probing-based matching is much more expensive due to the large number of HTTP requests sent for each interface. In addition, it needs global model for each domain and thus less scalable as an automatic generic solution for handling various domains of Web sources. Reference [Wu et al. 2004] pursues a clustering-based approach to discover 1:n matchings by exploring the “bridging” effect among query interfaces. However, its discovery of complex matchings essentially depends on a “hierarchical” interface extractor–

That is, the grouping of attributes (*e.g.*, the grouping of last name and first name) must be identified, in the first place, by the interface extractor (and not the matching algorithm). This "hierarchy-recognition" requirement makes interface extraction a very challenging task. In contrast, the DCM framework only requires an interface extractor to extract a query interface as a "flat" set of query conditions, instead of a hierarchy of attributes, which can thus be easily satisfied (*e.g.*, our recent work of automatic interface extraction [Zhang et al. 2004] is such an extractor). In fact, even with a simple "flat" extractor, it already introduces enough errors to impact the matching performance. In this article, we study such impact and propose an ensemble approach for maintaining the robustness of matching, which significantly extends the DCM framework (Figure 2), as described in our earlier work [He et al. 2004].

Our previous schema matching work, the MGS framework [He and Chang 2003], 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.

Second, in terms of our *techniques*: In this article, we propose a correlation mining view for the schema matching problem. Unlike previous correlation mining algorithms, which mainly focus on finding strong positive correlations [Agrawal et al. 1993; Tan et al. 2002; Omiecinski 2003; Lee et al. 2003; Brin et al. 1997], our algorithm cares not only positive but also 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.

In addition, as discussed in Section 5, our ensemble idea is inspired bagging classifiers [Breiman 1996]. However, our approach is different from bagging classifiers in three aspects: The setting, the techniques and the analytical modeling.

8. CONCLUSION

This article explores co-occurrence patterns among attributes 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. 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 difference leads to the introduction of a new correlation measure, H-measure, distinct from those proposed in previous work. Further, when connecting the DCM framework with automatic interface extractor to fully automate the matching process, we notice that noises in the interface extraction may significantly affect the matching result. To make the DCM framework robust against noisy schemas, we integrate it with an ensemble scheme by exploiting statistical sampling and voting. Our experiments validate the effectiveness of the correlation mining algorithm, the H-measure and the ensemble DCM framework.

REFERENCES

AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. N. 1993. Mining association rules between sets of items in large databases. In *SIGMOD 1993 Conference*.

ANDERSON, D. R., SWEENEY, D. J., AND WILLIAMS, T. A. 1984. *Statistics for Business and Economics (Second Edition)*. West Pub. Co.

BATINI, C., LENZERINI, M., AND NAVATHE, S. B. 1986. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys 18,* 4, 323–364.

BERGMAN, M. K. 2000. The deep web: Surfacing hidden value. Tech. rep., BrightPlanet LLC. Dec.

BORDA, J. C. 1781. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*.

BREIMAN, L. 1996. Bagging predictors. *Machine Learning 24,* 2, 123–140.

BRIN, S., MOTWANI, R., AND SILVERSTEIN, C. 1997. Beyond market baskets: generalizing association rules to correlations. In *SIGMOD 1997 Conference*.

BRUNK, H. D. 1965. *An Introduction to Mathematical Statistics*. New York, Blaisdell Pub. Co.

CHANG, K. C.-C., HE, B., LI, C., PATEL, M., AND ZHANG, Z. 2004. Structured databases on the web: Observations and implications. *SIGMOD Record 33,* 3, 61–70.

CHANG, K. C.-C., HE, B., LI, C., AND ZHANG, Z. 2003. The UIUC web integration repository. Computer Science Department, University of Illinois at Urbana-Champaign. http://metaquerier.cs.uiuc.edu/repository.

CHANG, K. C.-C., HE, B., AND ZHANG, Z. 2005. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR 2005 Conference*.

CHAUDHURI, S., GANJAM, K., GANTI, V., AND MOTWANI, R. 2003. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD 2003 Conference*.

DHAMANKAR, R., LEE, Y., DOAN, A., HALEVY, A., AND DOMINGOS, P. 2004. imap: Discovering complex semantic matches between database schemas. In *SIGMOD 2004 Conference*.

DIACONIS, P. AND GRAHAM, R. 1977. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society, Series B 39,* 2, 262–268.

DOAN, A., DOMINGOS, P., AND HALEVY, A. Y. 2001. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD 2001 Conference*.

DWORK, C., KUMAR, R., NAOR, M., AND SIVAKUMAR, D. 2001. Rank aggregation methods for the web. In *WWW 2001 Conference*.

FAGIN, R., KUMAR, R., AND SIVAKUMAR, D. 2003. Efficient similarity search and classification via rank aggregation. In *SIGMOD 2003 Conference*.

GOODMAN, L. AND KRUSKAL, W. 1979. *Measures of Association for Cross Classification*. New York: Springer-Verlag.

HE, B. AND CHANG, K. C.-C. 2003. Statistical schema matching across web query interfaces. In *SIGMOD 2003 Conference*.

HE, B. AND CHANG, K. C.-C. 2005. Making holistic schema matching robust: An ensemble approach. In *SIGKDD 2005 Conference*.

HE, B., CHANG, K. C.-C., AND HAN, J. 2004. Discovering complex matchings across web query interfaces: A correlation mining approach. In *SIGKDD 2004 Conference*.

HE, H., MENG, W., YU, C., AND WU, Z. 2003. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. In *VLDB 2003 Conference*.

IPEIROTIS, P. G., GRAVANO, L., AND SAHAMI, M. 2001. Probe, count, and classify: Categorizing hidden web databases. In *SIGMOD 2001 Conference*.

KEMENY, J. G. 1959. Mathematics without numbers. *Daedalus 88,* 571–591.

LANGLEY, P. 1995. *Elements of Machine Learning*. Morgan Kaufmann.

LEE, Y.-K., KIM, W.-Y., CAI, Y. D., AND HAN, J. 2003. Comine: Efficient mining of correlated patterns. In *Proc. 2003 Int. Conf. Data Mining*.

MADHAVAN, J., BERNSTEIN, P. A., AND RAHM, E. 2001. Generic schema matching with cupid. In *VLDB 2001 Conference*. 49–58.

MELNIK, S., GARCIA-MOLINA, H., AND RAHM, E. 2002. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE 2002 Conference*.

OMIECINSKI, E. 2003. Alternative interest measures for mining associations. *IEEE Trans. Knowledge and Data Engineering 15*, 57–69.

PORTER, M. The porter stemming algorithm. Accessible at http://www.tartarus.org/˜martin/PorterStemmer.

RAHM, E. AND BERNSTEIN, P. A. 2001. A survey of approaches to automatic schema matching. *VLDB Journal 10,* 4, 334–350.

SELIGMAN, L., ROSENTHAL, A., LEHNER, P., AND SMITH, A. 2002. Data integration: Where does the time go? *Bulletin of the Tech. Committee on Data Engr. 25,* 3.

TAN, P., KUMAR, V., AND SRIVASTAVA, J. 2002. Selecting the right interestingness measure for association patterns. In *SIGKDD 2002 Conference*.

WANG, J., WEN, J.-R., LOCHOVSKY, F., AND MA, W.-Y. 2004. Instance-based schema matching for web databases by domain-specific query probing. In *VLDB 2004 Conference*.

WU, W., YU, C. T., DOAN, A., AND MENG, W. 2004. An interactive clustering-based approach to integrating source query interfaces on the deep web. In *SIGMOD 2004 Conference*.

YOUNG, H. P. 1974. An axiomatization of borda's rule. *J. Economic Theory 9*, 43–52.

YOUNG, H. P. 1988. Condorcet's theory of voting. *American Political Science Review 82*, 1231–1244.

ZHANG, Z., HE, B., AND CHANG, K. C.-C. 2004. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *SIGMOD 2004 Conference*.

ZHANG, Z., HE, B., AND CHANG, K. C.-C. 2005. Light-weight domain-based form assistant: Querying web databases on the fly. In *VLDB 2005 Conference*.

Received Month Year; revised Month Year; accepted Month Year