PruSM: A Prudent Schema Matching Approach for Web Forms

Thanh Nguyen
University of Utah
thanhnh@cs.utah.edu

Hoa Nguyen
University of Utah
thanhhoa@cs.utah.edu

Juliana Freire
University of Utah
juliana@cs.utah.edu

ABSTRACT

There has been a substantial increase in the number of Web data sources whose contents are hidden and can only be accessed through form interfaces. To leverage this data, several applications have emerged that aim to automate and simplify the access to these data sources, from hidden-Web crawlers and meta-searchers to Web information integration systems. A requirement shared by these applications is the ability to understand these forms, so that they can automatically fill them out. In this paper, we address a key problem in form understanding: how to match elements across distinct forms. Although this problem has been studied in the literature, existing approaches have important limitations. Notably, they only handle small form collections and assume that form elements are clean and normalized, often through manual pre-processing. When a large number of forms is automatically gathered, matching form schemata presents new challenges: data heterogeneity is compounded with the Web-scale and noise introduced by automated processes. We propose PruSM, a prudent schema matching strategy the determines matches for form elements in a prudent fashion, with the goal of minimizing error propagation. A experimental evaluation of PruSM using widely available data sets shows that the approach effective and able to accurately match a large number of form schemata and without requiring any manual pre-processing.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases

General Terms

Algorithms, Experimentation

Keywords

Hidden Web, Web forms, schema matching.

1. INTRODUCTION

It is estimated there are millions of databases on the Web whose contents are hidden and are only exposed on demand, as users fill out and submit Web forms [12]. Several applications have emerged which attempt to uncover hidden-Web information and make it more easily accessible, including: metasearchers [8, 9], hidden-Web crawlers [2, 13], and Web information integration systems [5, 11, 20]. A common requirement for these systems is the ability to *understand* these forms, and in particular, to identify correspondences among elements in different forms. Web information integration systems and meta-searchers aim to provide a unified interface (a global schema) to a large number of data sources. By identifying correspondences among elements, it is possible to group elements with the same meaning, which leads to a more usable schema that can cover a potentially larger number of data sources. Hidden-Web crawlers automatically generate value assignments to the various form elements, and submit these to retrieve the hidden content. Having element correspondences available enables a crawler to gather sets of values, present in different forms, which can be used to fill out the forms. As an example, consider the forms in Figure 1, which are used to search for used cars. These forms use different representations as well as different labels to represent a given concept. In the first and second forms use labels `by Brand` and `Car Make` to represent the concept 'car make'. They also use different element types: a text field and a selection list, respectively. Thus, by identifying the correspondence between the two, a crawler could obtain a list of values in the second form to fill out the first one.

In this paper, we propose a new approach to the problem of *form schema matching*. Given a large set of Web forms, our goal is to automatically identify the correspondences (or matches) among elements in these forms. There are several challenges involved in deriving element correspondences for form schemas. As Figure 1 illustrates, there is a wide variability in how forms are designed, even within a well-defined domain. Different labels, including labels with no syntactic similarity (*e.g.,* `make` and `manufacturers`) are used to represent the same concept, while syntactically similar labels (*e.g.,* `manufacturer` and `year of manufacture`) are used to represent different concepts. Besides labels, element values are another source of similarity information that can be used to derive correspondences. But similar to labels, they can lead to mistakes. For example, `price` and `mileage` have similar values and yet represent different concepts. Co-occurrence statistics can also be useful to identify mappings [10, 18]. For example, by observing that *manufacturer* and *brand* co-occur with a similar set of labels (or attributes) but rarely co-occur together, it is possible to infer that they are synonyms. However, when used in isolation, attribute correlation can lead to incorrect matches. In particular, correlation matching scores (section 2.1.1) can be artificially high for rare attributes, since rare attributes seldom co-occur with (all)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.
Copyright 2010 ACM 978-1-4503-0099-5/10/10 ...$10.00.

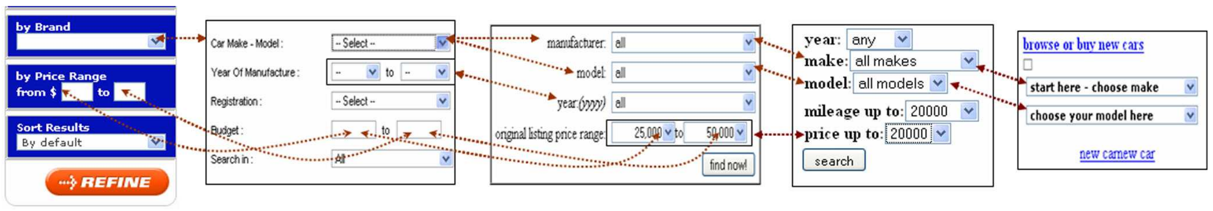

Figure 1: Matching form schemata in the Auto domain. Different labels are used to represent the same concept.

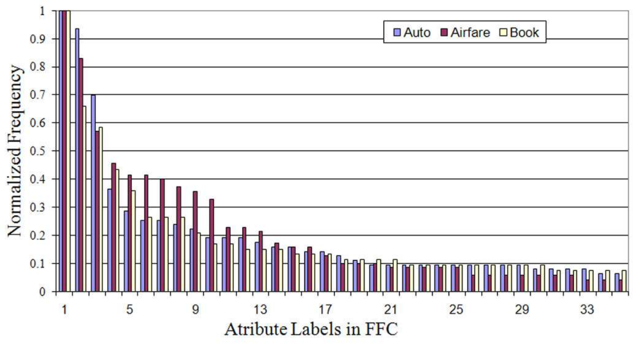

Figure 2: Label histogram in the FFC dataset

other attributes. Consequently, if only one source of similarity is used, be it labels or values, incorrect matches will be derived.

While solutions to problem of form-schema matching have been proposed in the literature [11, 20, 10, 18, 12, 19], they share an important limitation: they assume as input, a clean, manually-pre-processed set of forms. Consequently, they are not able to effectively handle large collections. Approaches [11, 7, 4, 20, 12, 19] which employ one matcher of a fixed combination of matchers may fail, since the effectiveness of a given matcher varies not only within domain but also for different forms within a domain. Holistic approaches [18, 10] benefit from having a large number of forms, but their performance depends on the availability of *manually* pre-processed data. Reliance on label simplification is problematic since manual simplification is not feasible for large collections, and automated techniques invariably lead to errors. Consider, for example, the concept *Model* in the used car domain. Some labels used for this concept include: `Choose vehicle model example mustang`, `What is your model`, `Please select a vehicle`. While `Model` is usually an important term, it is not important in `Model Year`. On the other hand, while the term `vehicle` is not important in `Choose vehicle model`, it is important in `Please select a vehicle`, where it denotes *make/model*.

Holistic approaches can also be negatively affected in the presence of *rare* labels, which occur often in different domains. Consider Figure 2 which shows the frequency of labels in three different form domains: Auto, Airfare and Books. In this histogram, we can observe a long tail, *i.e.,* few attributes have high frequency while many attributes have low frequency. Holistic approaches assume a Zipf-like distribution [10, 18], which can only be obtained by manually pre-processing the data and normalizing the labels.

Contributions. We propose PruSM, a new system for form-schema matching that combines multiple sources of information for similarity, and does so in a prudent fashion: by first deriving correspondences with high confidence, it minimizes error propagation. PruSM does not require forms to be pre-processed and it is robust to rare attributes. We also discuss experiments on two form collections which show that PruSM obtains higher accuracy than existing holistic approaches.

2. THE PRUSM SYSTEM

The high-level architecture of PruSM — Prudent Schema Matching is shown in Figure 3. Given a set of forms, the *Aggregation* module groups together form *elements* into *attribute sets*. The frequent attributes are used in *Matching Discovery* module. An attribute is considered *frequent* if its frequency is above the threshold T_c. Matching Discovery finds matches (both 1:1 and 1:n) among frequent attributes.

Aggregation. An important challenge in form-schema matching problem is the sparseness of values. Many form elements do not have any associated domain values, and even for the ones that do, the values of similar elements might be different in different forms. To attenuate this problem, PruSM aggregates similar elements (from different forms) and their values. It creates a set of *attributes* by grouping elements whose labels are the *same* after stemming and removing stop-words [1]. For instance, `Select a make` and `Select makes` are aggregated as an *attribute* with label `select make`. Our goal is to capture the underlying distribution for the values for the different concepts (or attributes). As we discuss below, aggregation is also fundamental for computing attribute correlations, which are based on set of elements.

2.1 Matching Discovery

As part of the prudent strategy, Matching Discovery (MD) aims to discover only matches that have high confidence, and to do so, it only considers frequent attributes. In addition, it combines the multiple sources of similarity in such a way that they re-inforce each other.

2.1.1 *Computing similarities*

Given two attributes (A_i, A_j), we quantify the similarity between them using three measures: label similarity, domain-value similarity and correlation.

Label Similarity. Because forms are designed for human consumption, labels are descriptive and are often an effective means for identifying similar elements. We define the *label similarity* between two attributes (A_i, A_j) as the cosine distance [1] between the *term vectors* for their labels:

$$lsim(A_i, A_j) = cos(l_i, l_j) \quad (1)$$

Each term is associated to a weight that captures its importance. To capture the importance of terms, we use both term frequency (TF) [1] and the single-token frequency (STF) [14]. The latter distinguishes between labels that frequently appear alone and are thus likely to be important (*e.g.,* `Make`, `Model`), and labels that only appear together with other labels, and are thus unlikely to represent an important concept (*e.g.,* `Please`, `Choose`).

Domain-Value Similarity. Recall that elements with the same label are clustered together. Before computing the domain similarity between two attributes, we aggregate all the domain values for each element in the attribute cluster. Given an attribute cluster A_k, we

[1]http://members.unine.ch/jacques.savoy/clef/englishST.txt

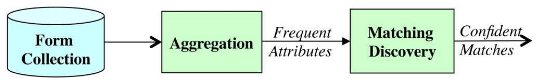

Figure 3: The high-level architecture of PruSM

build a vector that contains the occurrences of all values associated with the label l for A_k and their frequencies:

$$D_k = \bigcup_{i=1..n} (v_i : frequency)$$

Given two attribute clusters A_i and A_j, the cosine distance is then used to measure the similarity between their corresponding value vectors:

$$dsim(A_i, A_j) = cos(D_i, D_j) \quad (2)$$

Correlation. By holistically analyzing a set of forms, we can leverage an implicit source of similarity information: attribute correlation. To compute correlation, In PruSM, we use X/Y measures [18] which are defined as follows:

$$X(A_p, A_q) = \begin{cases} 0 & \text{if } A_p, A_q \subset f \\ \frac{(C_p - C_{pq})(C_q - C_{pq})}{(C_p + C_q)} & \text{otherwise} \end{cases} \quad (3)$$

$$Y(A_p, A_q) = \frac{C_{pq}}{min(C_p, C_q)} \quad (4)$$

C_p, C_q, C_{pq} correspond to the number of forms which contain A_p, A_q and both A_p, A_q. The matching score X captures negative correlation while the grouping score Y captures positive correlation. The intuition comes from the fact that synonym attributes are semantic alternatives and rarely co-occur in the same form interface —they are negatively correlated (*e.g., Make* and *Brand*). On the other hand, grouping attributes are semantic complements and often co-occur in the same form interfaces —they are positively correlated (*e.g., First name* and *Last name*).

2.1.2 Prudent Matcher

The goal of Prudent Matcher is to identify matches with high confidence. We create (independent) *matchers* [17, 16, 7, 6] for each feature: label similarity, domain-value similarity and correlation. Although these features are obtained at the level of set of elements, using them in isolation is still insufficient and leads to error propagation. We then define a *prudent matcher*, which consists of a set of constraints over these independent matchers. A prudent match is *valid* if

X(A_p, A_q) $> T_{Matching\_score}$ AND
[$dsim(A_p, A_q) > T_{dsim}$ OR $lsim(A_p, A_q) > T_{lsim}$].

The prudent matcher is a composite matcher. It is simple yet comprehensive because it incorporates both visible similarity information (*e.g.,* label similarity and value similarity) and latent information (attribute correlation) at a high level of set of elements. The intuition behind the prudent matcher is that the different features re-inforce each other. For example, even when two attributes A_p and A_q have a high correlation score, they do not provide a *good* match unless additional evidence is available.

2.1.3 Constructing Matches

We construct a set of confident matches M by iteratively choosing highest negatively correlated attribute pairs which pass the prudent matcher, and deciding whether a match should be added to an existing set or whether a new matching component should be created.

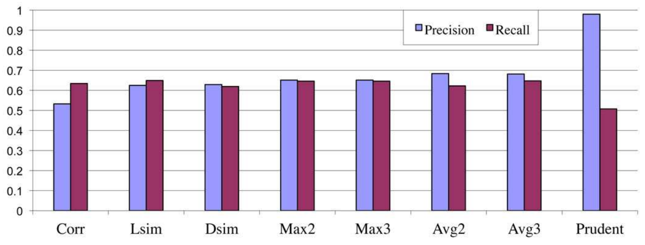

Figure 4: Precision and recall for different strategies to combine similarity in the matching discovery module of PruSM

3. EXPERIMENTAL EVALUATION

3.1 Experimental Setup

We evaluate the PruSM performance over two public form collections: TEL8 and FFC. TEL8[2] consists of a set of manually collected and pre-processed forms. FFC contains forms that were automatically gathered by a focused crawler [3], therefore—it is large and more heterogeneous than TEL8. For both data sets, we manually extracted the form-element labels. As shown in Figure 2, there is a wide variability in the frequency distribution of these labels, especially the large number of rare attributes in the long tail for the FFC collection.

We compare PruSM against HSM [18] using these two data sets. We use our own implementation of HSM with a frequency threshold T_c=5%. In addition, we use the element labels as is—no syntactic merging was applied.

Effectiveness measures. To evaluate the effectiveness of PruSM, we use precision, recall and F-measure [1]. Since there are many matches, we measure the average precision, recall and F-measure according to the sizes of each match.

3.2 Preliminary results

Prudent Matcher. We compare the prudent matcher against other matchers including: singular matchers (correlation, lsim, dsim); combination of these matchers *e.g.,* linear combination of lsim and dsim (Avg2) or lsim, dsim and correlation (Avg3) or the maximum value among lsim and dsim (Max2) or lsim, dsim and correlation (Max3). Figure 4 shows that the prudent matcher can avoid a large number of incorrect matches and obtains substantially higher precision than the others, which is a key requirement for Matching Discovery—to avoid error propagation.

Effectiveness of Matching Discovery. Figure 3.2 shows the accuracy values of HSM and PruSM_MD. HSM has lower accuracy because we use a low frequency threshold T_c and use the labels as they are. In particular, HSM has low accuracy in heterogeneous domains like FFC-Book and higher accuracy in 'clean' domains like TEL8-Book. For both data sets, and in all domains, the precision, recall and F-measure values of PruSM are higher than those for HSM because it can obtain both syntactic and semantic matches. For example, PruSM can identity many variations of *auto make* including `car make`, `select vehicle`, `manufacture`, `choose a make`, etc. The gains in F-measure of PruSM_MD compared to HSM vary between 9% and 30% in TEL8, and between 33% and 39% in FFC. Smaller improvements are observed in clean domains (*e.g.,* TEL8 Book—9.1%, and TEL8 Airfare—9.3%), where data is cleaner and HSM is expected to perform well. Bigger improvements are observed in more heterogeneous domains of FFC (*e.g.,* FFC Book—37.9% and FFC Auto—39.4%).

[2]http://metaquerier.cs.uiuc.edu/repository

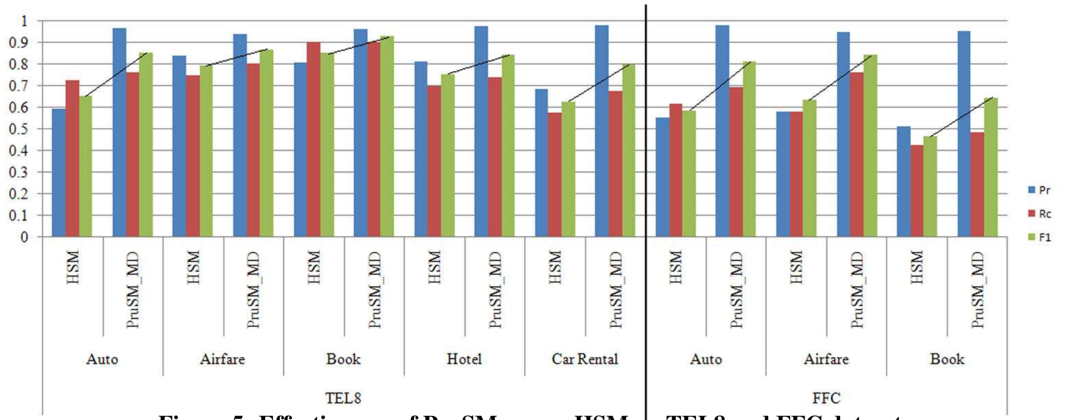

Figure 5: Effectiveness of PruSM versus HSM on TEL8 and FFC dataset

4. RELATED WORK

Several approaches have been proposed for matching forms [19, 20, 11, 10, 10, 18]. *Clustering-based* approaches [20, 11, 4] combine visible features of individual form elements to define a similarity function between two elements in different forms. Besides requiring *clean* data as input, most of these approaches are only effective for a small number of forms. To identify *synonyms*, they use Wordnet [11] which is impossible to identify domain-specific synonyms (like `Vehicle` and `Make`); or leverage the domain values [15, 17], which is insufficient for attributes that range over a sparse domain, or attributes that are not associated with values.

Holistic approaches [10, 18] benefit from considering a large number of schemas. DCM [10] exploits the "apriori" property to discover all possible positively and negatively correlated groups. HSM [18] uses a correlation-based greedy algorithm to discover element synonyms. However, similar to clustering-based approaches, they also require clean and normalized data which limits their scalability. They also ignore *rare* attributes—when rare attributes are present, their performance decreases significantly [10, 18].

Although two-phase matching has been used for matching ontology and database schemata [11, 7, 4], they assume certain matchers are strong and combine them in a fixed manner. Given the Web data heterogeneity, using a fixed combination of matchers at the level of individual elements is not sufficient. PruSM comprehensively combines meta-features in a prudent *matcher* and also utilizes strong *matches* as additional constraints to reconcile weaker and uncertain matches.

5. CONCLUSIONS

In this paper, we proposed PruSM, an *automated* schema matching approach that is effective for large and heterogenous forms collections. PruSM applies schema matching in a prudent fashion, focusing on high-frequency attributes which provide statistically significant evidence, and combines multiple sources of similarity in a such a way that they re-inforce each other.

6. REFERENCES

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

[2] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In *In SBBD*, pages 309–321, 2004.

[3] L. Barbosa and J. Freire. An adaptive crawler for locating hidden-web entry points. In *WWW*, pages 441–450, 2007.

[4] N. Bozovic and V. Vassalos. Two-phase schema matching in real world relational databases. In *ICDE Workshops*, pages 290–296, 2008.

[5] K. C.-C. Chang, B. He, and Z. Zhang. Toward Large-Scale Integration: Building a MetaQuerier over Databases on the Web. In *CIDR*, pages 44–55, 2005.

[6] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. *SIGMOD Rec.*, 30(2):509–520, 2001.

[7] A. Gal. Managing uncertainty in schema matching with top-k schema mappings. pages 90–114, 2006.

[8] Google Base. http://base.google.com/.

[9] L. Gravano, H. García-Molina, and A. Tomasic. Gloss: text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.

[10] B. He and K. C.-C. Chang. Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Trans. Database Syst.*, 31(1):346–395, 2006.

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

[12] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffery, D. Ko, and C. Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR*, pages 342–350, 2007.

[13] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Y. Halevy. Google's deep web crawl. *PVLDB*, 1(2):1241–1252, 2008.

[14] H. Nguyen, T. Nguyen, and J. Freire. Learning to extract form labels. *Proc. VLDB Endow.*, 1(1):684–694, 2008.

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

[16] A. D. Sarma, X. Dong, and A. Halevy. Bootstrapping pay-as-you-go data integration systems. In *In Proc. of SIGMOD*, 2008.

[17] P. Shvaiko and P. Shvaiko. A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171, 2005.

[18] W. Su, J. Wang, and F. Lochovsky. Holistic query interface matching using parallel schema matching. In *Advances in Database Technology - EDBT*, pages 77–94, 2006.

[19] J. Wang, J.-R. Wen, F. L, and W.-Y. Ma. Instance-based schema matching for web databases by domain-specific query probing. In *VLDB*, pages 408–419, 2004.

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