Duplicate Identification in Deep Web Data Integration

Wei Liu[1], Xiaofeng Meng[2], Jianwu Yang[1], and Jianguo Xiao[1]

[1] Institute of Computer Science & Technology, Peking University,
Beijing, China
[2] School of Information, Renmin University of China,
Beijing, China
gue1976@gmail.com, xfmeng@ruc.edu.cn,
{yangjianwu,xjg}@icst.pku.edu.cn

Abstract. Duplicate identification is a critical step in deep web data integration, and generally, this task has to be performed over multiple web databases. However, a customized matcher for two web databases often does not work well for other two ones due to various presentations and different schemas. It is not practical to build and maintain C_n^2 matchers for *n* web databases. In this paper, we target at building one universal matcher over multiple web databases in one domain. According to our observation, the similarity on an attribute is dependent of those of some other attributes, which is ignored by existing approaches. Inspired by this, we propose a comprehensive solution for duplicate identification problem over multiple web databases. The extensive experiments over real web databases on three domains show the proposed solution is an effective way to address the duplicate identification problem over multiple web databases.

Keywords: duplicate identification, deep web data integration, web database.

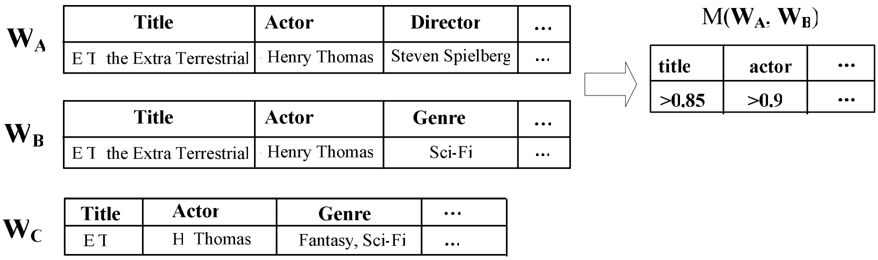
1 Introduction

Survey[6] revealed deep web is being the largest information depository on the web. Deep web data integration is becoming a hot area for both research and industry. There is often high redundancy among web databases, so it is a necessary step to match duplicates for data cleaning or further applications, such as price comparison services and the information merging.

Duplicate identification(a.k.a. de-duplication, record linkage, etc.) is the process of identifying different representations of one entity, which is always a challenging task in heterogeneous data sources integration. To the best of our knowledge, lots of solutions have been proposed to address this issue[7]. However, most of them focus on the problem only between two sources. Due to the large scale of deep web, lots of web databases are integrated in practice. As a result, C_n^2 matchers have to be built for *n* web databases if traditional approaches were applied.

Fig.1 shows three movies records from web databases W_A, W_B and W_C. M(W_A, W_B) is the customized duplicate matcher for W_A and W_B, and an example rule requires *title* similarity of two duplicate records is larger than 0.85. The threshold is high due to the

L. Chen et al. (Eds.): WAIM 2010, LNCS 6184, pp. 5–17, 2010.



W_A

Title	Actor	Director	...
E T the Extra Terrestrial	Henry Thomas	Steven Spielberg	...

W_B

Title	Actor	Genre	...
E T the Extra Terrestrial	Henry Thomas	Sci-Fi	...

W_C

Title	Actor	Genre	...
E T	H Thomas	Fantasy, Sci-Fi	...

M(W_A, W_B)

title	actor	...
>0.85	>0.9	...

Fig. 1. An example to illustrate the limitation of most traditional approaches

full *titles* given by W_A and W_B. But M(W_A, W_B) is not applicable for W_C and W_B because the *title* similarity is smaller than the threshold though the two records are matched in fact. More importantly, the new shared attribute, *genre*, cannot be handled by M(W_A, W_B). As a result, M(W_A, W_C) and M(W_B, W_C) have to be built. Several works[5][13] have realized this fact, and they try to select the best matching technique or combine multiple matching algorithms to improve matching performance. But it is a challenging problem to build and maintain the library of matching algorithm and select the optimal one.

In this paper, we study the problem of duplicate identification over multiple web databases. There are a large number of web databases in one domain and usually lots of web databases are integrated, and it is not practical to build and maintenance lots of matchers. The proposed approach is based on two interesting observations. First, the presentation variations of an attribute are finite, which means an instance of it can be transformed into other forms according to some variation rules(e.g. the person name "Jim Gray" has limited variations, such as "J. Gray", "Gray J.".). Second, there exists similarity dependency among attributes, i.e. the similarity of one attribute can be improved by the similarities of other attributes. For example, if having known the actors and the publication dates of two movie records are same, we are more certain the titles are same too. However, previous works only calculate the attribute similarity independently without considering their dependency.

In summary, the contributions of this paper are present as follows. First, we believe this is the first attempt to build one matcher for multiple web databases, which is an interesting issue in deep Web data integration. Second, we identify the similarity dependencies among attributes, and proposed an inference-based method to improve the record similarity by exploiting the similarity dependencies. Third, an efficient approach is proposed to building the universal matcher, which can greatly reduce the cost of manual labeling.

The rest of this paper is organized as follows. Section 2 introduces the variation rules to handle various representations of attribute values, and further give the uniform representation of record similarity. An inference-based method to improve the record similarity is described in Section 3. Section 4 introduces the approach of building the universal matcher. Section 5 presents the experiments. Section 6 contains the related works. Section 7 is the conclusion.

2 Uniform Representation for Attribute Similarity and Record Similarity

This section first presents the variation rules of attributes, and then gives the uniform representation of record similarity based on the variation rules.

Table 1. Classification of variation rules

Classification	Rule	Example
Character Level Rules	*Prefix*	"Johnson" vs. "J."
	Suffix	"2008" vs. "08"
	Prefix+Suffix	"Dept" vs. "Department"
	Plural	"university" vs. "universities"
Token Level Rules	*Prefixes Concatenation*	"Caltech" vs. "California Institute of Technology"
	Acronym	"UCSD" vs. "University of California, San Diego"
	Subsequence	"Java 2 Bible" vs. "Java 2 (JSEE 1.4) Bible"
	Rearrangement	"2/18/2008" vs. "18/2/2008"
Semantic Level Rules	*Synonymy*	"Automobile" vs. "Car"
	Hypernymy/Hyponymy	"Vehicle" vs. "Car"

2.1 Variation Rules

Different representations would be found for the same object from different web databases. A number of methods have been developed to calculate the attribute similarity by assigning a 0-1 real number. Such real-value based approaches are not applicable for our task , just like the example shown in Fig.1. Therefore, a uniform similarity metric is needed to represent the attribute similarity. In fact, the presentations of an attribute follow finite variation rules. We summarize and classify all the observed variation rules in Table 1, and the examples are also given.

For each attribute, one or several rules are assigned to it by domain experts. For instance, four rules, "*Prefix*", "*Acronym*", "*Subsequence*" and "*Rearrangement*", are assigned to the person name related attributes(e.g. "actor" and "director") in movie domain. Though being a manual task, the assignments are universal and stable on domain level. In practice, assigning rules to about 25 popular attributes in one domain is enough because the "rare" attributes have little contribution.

2.2 Representations of Attribute Similarity and Record Similarity

An important characteristic of our approach is using three logic values (Yes, Maybe and No) instead of 0-1 real values to represent attribute similarity. Different to the probabilistic-based approaches, the three logic values are based on the variation rules defined below (suppose v_1 and v_2 are two compared values):

(1) YES (*Y*). If v_1 and v_2 are absolutely same, their similarity is "YES".
(2) MAYBE (*M*). If v_1 and v_2 are not same, but $v_1$could be transformed into v_2 using the assigned variation rules. For example, “John Smith” can be transformed into "S. John" with “*Rearrangement*” and “*Prefix*” rules. In this situation, it is uncertain whether v_1 and v_2 are the same semantic, and “MAYBE” is labeled.
(3) NO (*N*). If v_1 cannot be transformed into v_2 by applying the assigned rules, “NO” is labeled, such as “John Smith” and “S. Jensen”.

Based on the three logic values, attribute similarity can be represented as “Y”, “M” or “N”. And record similarity can be represented as $S(r_1, r_2) = \{<a_1, s_1>,<a_2, s_2>,\ldots,<a_k, s_k>\}$, where a_i $(1\leq i\leq k)$ are their shared attributes, and $s_i \in \{Y, M, N\}$Table 2 shows an example to illustrate the representation of record similarity.

Table 2. An example of record similarity

	Title	**Actor**	**Director**	**Publication date**
r_1	E.T.: The Extra-Terrestrial	Henry Thomas	S. Spielberg	1982
r_2	E.T.	Henry Thomas	Steven Spielberg	1982
$S(r_1, r_2)$	*M*	*Y*	*M*	*Y*

3 Record Similarity Improvement

“M” is the obstacle to duplicate identification, and it must be “Y” or “N” in fact. Hence, if “M” is transformed into “Y” or “N” with more evidences, duplicates can be identified more accurately. Existing approaches do not catch sight of the inherent similarity dependency among attributes. In this section, we first propose the concept of attribute similarity dependency and then present a novel method to improve record similarity by using Markov Logic Networks to exploit the attribute similarity dependency.

3.1 Attribute Similarity Dependency

Table 3. Several examples to illustrate the attribute similarity dependency

First-logic Formulas	**Weights(importance)**
Y(actor) ∧ Y(publication date) ∧ M(director) → Y(director)	3.872
Y(director) ∧ Y(publication date) ∧ M(title) → Y(title)	3.551
Y(director) ∧ Y(actor) ∧ M(genre) → Y(genre)	2.394

Previous approaches do not give insight into attribute similarity dependency. Actually, there exists potential similarity dependency among attributes. For example, it is difficult to determine the titles of the two records in Table 2 are same no matter which current similarity function is applied. But intuitively, we will be sure the titles are same if having known both actors and publication dates are same.

Definition. *Attribute similarity dependency.* We say the similarity on attribute a_0 is dependent of those on attributes $\{a_1,\ldots,a_m\}$ *iff* $\mathrm{P}(a_0 = s_0) \neq \mathrm{P}(a_0 = s_0 | a_1 = s_1, \ldots, a_m = s_m)$ Where $s_i \in \{\mathrm{Y}, \mathrm{M}, \mathrm{N}\}$ denotes the similarity on a_i. Otherwise, the attribute a_0 is independent of attributes $\{a_1,\ldots,a_m\}$.

In fact, this is an interactive and propagating process. Table 3 shows three examples which are represented in form of the first-logic formulas to illustrate this process. Formula 1 means the similarity M on *director* can be improved to Y if the similarities on *actor* and *publication date* are Y. As we have seen, Formula 1 and Formula 2 illustrate the interactive process, while Formula 1 and Formula 2 illustrate the propagating process. To discover attribute similarity dependency, we employ Markov Logic Networks(MLNs)[11] to model the graph-like dependency. The basic concept of MLNs is introduced first, and then the method of exploiting the attribute similarity dependency using this model is presented.

3.2 Markov Logic Networks

MLNs are a combination of probabilistic graph model and first-order logic to handle the uncertainty in the real world. In a first-order logic, if a world violates one constraint it will have probability zero. MLNs soften the constraints of the first-order logic. If a world violates one formula it is less probable, but not impossible. Thus, MLNs is a more sound framework since the real world is full of uncertainty and violation. In MLNs, each formula has an associated weight to show its importance: higher the weight is, the greater the difference in log probability between a world that satisfies the formula.

MLNs is a template for constructing Markov Network [10]. With a set of formulas and constants, MLNs define a Markov network with one node per ground atom and one feature per ground formula. The probability of a state *x* in such a network is:

$$P(X = x) = \frac{1}{\mathrm{Z}} \prod_i \emptyset_i(x_{\{i\}})^{n_i\{x\}} \tag{1}$$

where Z is a normalization constant, $n_i(x)$ is the number of true groundings of formula F_i in x, $x_{\{i\}}$ is the state of the atoms appearing in F_i, and $\emptyset_i(x_{\{i\}}) = e^{wi}$, w_i is the weight of F_i. Eq. (1) is a generative MLNs model, that is, it defines the joint probability of all the predicates. More details of MLNs is discussed in [11].

3.3 Automatic Formulas Generation

The formulas of first-order logic are constructed using four types of symbols: constants, variables, functions, and predicates. The predications are the three logic values {Y, M, N}. The constants are the attributes of one domain. In existing works(e.g. [19]), the formulas of MLNs are written manually by experts. It is time-consuming and difficult to find the important formulas. Alternatively, we generate formulas automatically based on the following formula templates.

Thus, the formula templates in a given domain can be represented as the following form:

$$\forall a_0, a_1 \dots, a_{\mathrm{m}}\ \ \mathrm{Y}(a_1) \wedge \mathrm{Y}(a_2) \dots \wedge \mathrm{Y}(a_{\mathrm{m}}) \wedge \mathrm{M}(a_0) \rightarrow \mathrm{Y}(a_0) \tag{2}$$

where a_{i} is an attribute, m is the number of attributes are involved in this formula. The example formulas shown in Table 2 are generated with the formula templates, and their weights are trained with MLNs. A large number of formulas will be produced by Eq. (2)

The dependencies of two attributes are found first, then the dependencies of i attributes are found by extending the dependencies of i-1 attributes. The whole process of the formulas generation includes two stages. In the first stage, all formulas involving two attributes, such as $\forall \mathrm{x}_1\ \mathrm{A}_{\mathrm{i}}(\mathrm{x}_1) \wedge \mathrm{A}_{\mathrm{j}}(\mathrm{M}) \rightarrow \mathrm{A}_{\mathrm{j}}(\mathrm{Y})\ (\mathrm{i} \neq \mathrm{j})$, are exhausted as the initial formula set, and further, weights are assigned by MLNs through training. In the second stage, high-weight(>1) formulas are selected, and the two formulas whose query predicates are same are merged into one two-evidence-predicate formula. And further three-evidence-predicate formulas are generated by merging two-evidence-predicate formulas and one-evidence-predicate formulas.

4 Efficient Approach to Building Universal Matcher

In the traditional way, the training set for n web databases is times the training set for two web databases. It is not practical to label such large a training set. To reduce the labeling cost, this section proposes an efficient approach for building the universal matcher for n web databases.

The basic idea of our approach is that, given n web databases $\{W_1, W_2, \dots, W_{\mathrm{n}}\}$, the initial matcher is built for W_1 and W_2, and then it is evolved into a new one when W_3 is incorporated. The evolvement process stops until all web databases have been incorporated. The flowchart of building the universal matcher is shown in Fig. 2, which consists of three stages. The rest of this section introduces them in turn.

4.1 Training Set Generation

Two types of training sets are generated: labeled training set for Stage II and unlabeled training set for Stage III. The labeled training set is the record pairs that have been labeled as matched or not. For each record pair, one record is from W_1 and the other is from W_2. For each record pair in the unlabeled training set, two records are from different web databases except W_1 and W_2. All the record pairs have been processed by the uniform representation component and the record similarity improvement component and transformed into the form of the example shown in Table 1.

4.2 Initial Matcher Building

An initial matcher is built with the labeled training set for W_1 and W_2. We still use MLNs to build the initial matcher. The evidence predicates are the shared attributes of the two web databases, and the only query predicate is the match decision for the record pairs. The formulas can be represented as follows:

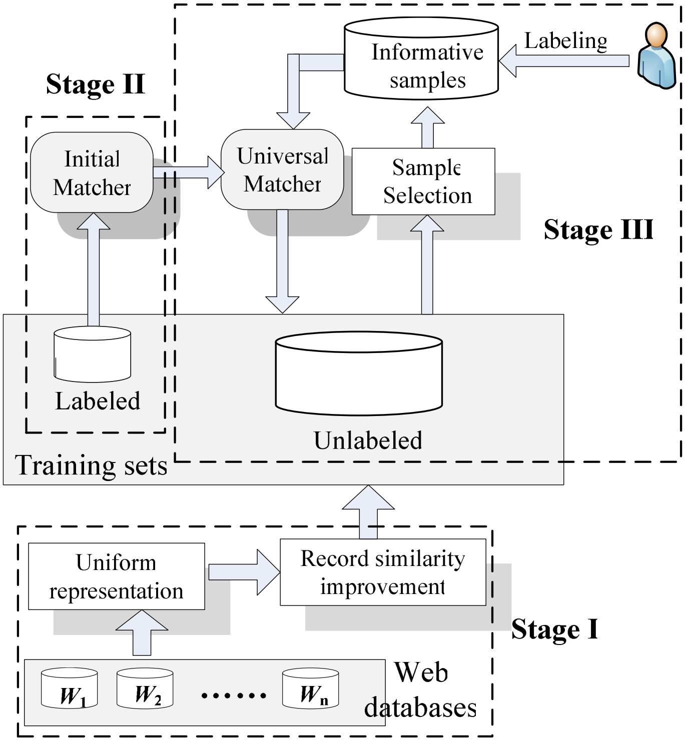


Fig. 2. Flowchart of building universal matcher

$$\forall x_1, x_2 \ldots, x_m, r_1, r_2 \quad S_1(a_1) \wedge S_2(a_2) \ldots \wedge S_m(a_m) \rightarrow \mathrm{Match}(r_1, r_2) \tag{3}$$

where a_i is an attribute of the two records shared, $S_i \in \{Y, M, N\}$. The record pair (r_1,r_2) is matched if Match(r_1,r_2) is true, otherwise not matched. Considering the efficiency, the low-weight formulas (say, less than 0.3) can be pruned at this stage because they have little contributions to the performance.

4.3 Matcher Evolving

When the initial matcher has been built, it still cannot get a better performance over all web databases because the labeled training set is only from W_1 and W_2. We propose an evolving strategy to reduce the labeling cost. As shown in Fig. 3, the evolving strategy is a cycle process. In each round, some informative samples are selected automatically by the current matcher and are labeled. And then the matcher is improved with the labeled record pairs. The process stops until the matcher cannot be improved anymore.

In this process, the key issue is how to automatically select the informative samples from the unlabeled training set for labeling. The informative samples can be classified into two types. The first type is the record pairs that contain new attributes, and the second type is the ones that cannot be determined with a high probability. The first type are easy to detect. For the second type, a sample is selected if its probability assigned by the current matcher is in the range $[1-\alpha,\alpha]$ $(0.5 < \alpha < 1)$, which means this sample cannot be determined by the current matcher with high probabilities. To avoid too many samples are labeled in each round, at the *t*+1th round is computed using at the *t*th round. The matcher at the first round is the initial matcher obtained in Stage II, and at the second round is set as 0.65 based on the experience. The formula for computing at the *t*+1th round is:

$$\alpha_{t+1} = \alpha_t + \frac{\sum_{s_i^+ \in TS_{lab}}(P_t(s_i^+)-P_{t-1}(s_i^+)) + \sum_{s_i^- \in TS_{lab}}(P_{t-1}(s_i^-)-P_t(s_i^-))}{|TS_{lab}|} \quad (4)$$

where α_t are α at the *t*th round, TS_{lab} is the current labeled training set (because some labeled samples are appended to TS_{lab} in each round), and are the positive samples and the negative samples of TS_{lab} respectively, P_t and P_{t-1} are the probabilities of an sample predicted by the matcher at the *t*th round and the matcher at the *t*-1th round respectively. When $\alpha_{t+1} < \alpha_t$, the evolving process stops, which means the current matcher be improved when more samples are labeled.

5 Experiments

5.1 Experiment Setup

Our data set includes three popular domains: movie, book, and research paper.
For each domain, we select five popular web databases. Table 4 shows the details about the date set. The records in the data set are extracted by the customized

Table 4. The Data Set used in the Experiments

Domain	Web database	URL	The number of records
Movie	IMDB	uk.imdb.com	62793
	Yahoo	movies.yahoo.com	10288
	the-numbers	www.the-numbers.com	5486
	aol	movies.aol.com	6541
	listentoamovie	power.listentoamovie.com	554
Book	Amazon	www.amazon.com	18623
	abebooks	www.abebooks.com	10039
	Bookpool	www.bookpool.com	3881
	collinsbooks	www.collinsbooks.com.au	2020
	Chapters	www.chapters.indigo.ca	1051
Research paper	DBLP	dblp.uni-trier.de/xml	38738
	Libra	libra.msra.cn	3459
	ACM	portal.acm.org	2738
	citeseer	citeseer.ist.psu.edu	1207
	springer	www.springerlink.com	189

wrappers for the web databases. The whole data set is divided into two parts averagely. One is as the training set, and the other is as the test set.

We use traditional *precision*, *recall* and *F-measure* metrics to evaluate our solution. precision is defined as the ratio of the total number of correctly matched record pairs and the total number of matched record pairs, and *recall* is defined as the total number of correctly matched record pairs and the total number of actual matched record pairs. *F-measure* is the harmonic mean of *precision* and *recall.*

5.2 Performance

The performance of our approach is evaluated on the three domains respectively, and the order of web databases coincides with the order shown in Table 2. We also implemented a rule-based matcher as the baseline for comparison, i.e., , one rule like that in Fig. 1 for each domain is trained. For the attributes of person name and organization, the *Smith*-Waterman *distance* algorithm [15] is adopted. For other attributes, the traditional edit distance is adopted. We use the popular tool Weka to learn the thresholds of the attributes. The experimental results are shown in Fig. 5. We explain the experimental results on two aspects. (1) The performance of our approach is much better than the rule-based approach on all the three measures. This proves our approach is superior than existing solutions to perform duplicate identification for multiple web databases. (2) The performance in research paper domain is better than those in other two domains. The main reason is that spelling errors in research paper domain are much less than the other two domains. Since once a spelling error occurs, the attribute similarity is “N”, and the duplicates this record will have little chance to be matched.

Table 5. The performances comparison between the rule-based approach and our approach

		Our approach	**rule-based approach**
precision	Book	92.7%	57.3%
	Movie	93.5%	68.7%
	Research paper	96.8%	78.4%
recall	Book	85.3%	42.9%
	Movie	88.4%	59.1%
	Research paper	91.2%	73.5%
F	Book	88.8%	48.7%
	Movie	90.9%	63.5%
	Research paper	93.4%	75.9%
AVG-F		91.0%	62.7%

5.3 Evaluation of Record Similarity Improvement

The goal of this experiment is to evaluate the contribution of record similarity improvement to the performance of the matcher. We turned off the record similarity improvement component is and then carry out the experiment again. As it can be seen from Fig. 3, the performance can improved greatly by the record similarity improvement component in all the three domains(especially book domain).

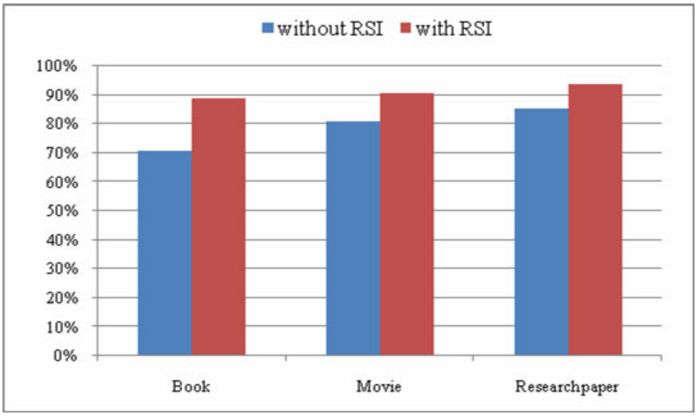


Fig. 3. Evaluation results of record similarity improvement(RSI) on *F measure*

5.4 Labeling Cost

In this part, we evaluate the labeling cost during the evolving process. Fig. 4 shows the labeling costs on the three domains. y axis refers to the number of labeled samples. From the experimental results indicate that the number of labeled samples drops significantly as the evolving rounds increasing. When the evolving rounds are more than 9, no samples need labeling for research paper domain (the evolving process stops), while only about 100 samples are required to label for other two domains. In addition, the curve of book domain is not as regular as those of other two domains. This phenomenon is caused by two reasons. The first reason is spelling errors occurs in this domain, and it is useless for improving the matcher even if a spelling-error record pair is labeled,. Second, the involved attributes of unlabeled training set contain some new ones compared to those of labeled training set, which makes more samples be labeled at the first time of the evolving process.

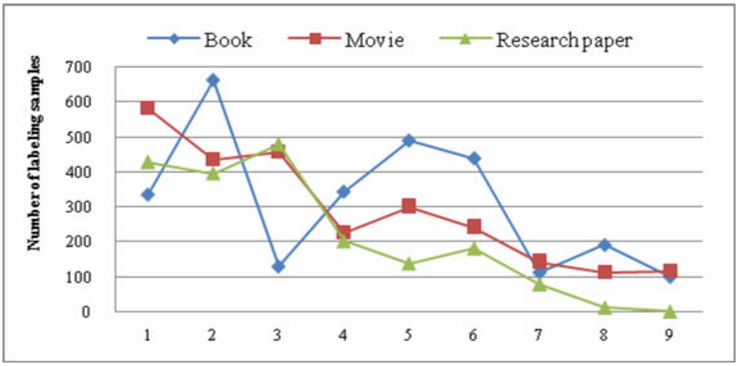


Fig. 4. Trend of the labeling cost during the evolving process

5.5 Scalability

Our ultimate goal is “one matcher, one domain”, i.e. only one duplicate identification matcher is competent for one domain. In this way, no training is needed when new web databases are incorporated. To verify this hypothesis, we use the record pairs from W_1, W_2, W_3 and W_4 as the training set to build the matcher, and the test set is the record pairs in which one record is from W_5 and the other is from W_1, W_2, W_3 or W_4. In this way, W_5 can be viewed as the new incorporated web database. The experimental results in Fig. 5 are very close to those in Table 5. This indicates that the matcher can still achieve a good performance without training when a new web database(W_5) is incorporated.

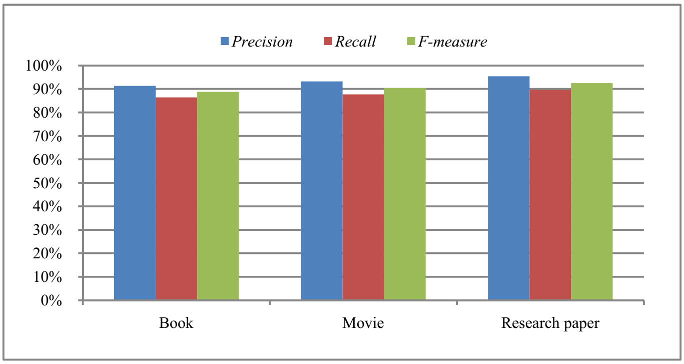


Fig. 5. The experimental results on robustness

6 Related Work

Duplication identification has been a thriving area of data integration surveyed in [9] and [7]. Previous researches mainly focused on similarity functions. They can be classified into two kinds according to the tasks they engaged in.

Attribute similarity. The most common reason of mismatches is the presentation variations of shared attributes. Therefore, duplicate detection typically relies on string comparison techniques to deal with presentation variations. Multiple methods have been developed for this task. Edit distance and its variations [9], jaccard similarity, tf-idf based cosine similarity [10] are the most popular similarity functions. Some of them are semantic-specific, such as the Jaro distance [6] and Jaro Winkler distance [7] for person names, and functions used by tools such as Trillium [8] for addresses.

Duplicate identification. The records usually consist of multiple attributes, making the duplicate detection problem more complicated. There are multiple techniques for duplicate detection, such as Bayesian-inference based techniques[16], active-learning based techniques[2], distance based techniques[4], rule based approaches[8], etc. A recent trend is to investigate algorithms that compute the similarity join exactly. Recent advances include inverted index-based methods[14], prefix filtering-based techniques[3,18], and signature-based methods[1]. Most of them focused on designing an appropriate similarity function. However, no similarity function can conciliate all presentation variations of an entity(record). Bayesian-inference-based techniques are the most similar approach with ours. They also represent record similarity as a vector $\{x_1,x_2,\ldots,x_n\}$ on the shared attributes, and 0 and 1 are used to denote the similarity on x_i. So assigning 0 or 1 to x_i properly is the basis of this approach. But in the context of deep web integration, many web databases will incur more presentation variations, which makes this task very challenging and further impacts on the accuracy of inference. Our approach supplements "Maybe" to accept the uncertainty and eliminate the uncertainty as far as possible by exploring the similarity dependency among attributes.

Overall, our solution includes both the two tasks above, and the closely coupled solutions are proposed for them respectively. The main differences of our solution and previous works are on two aspects. First, we put forward three logic values coupled with predefined variation rules instead of traditional similarity functions. Second, we first discover and exploiting the similarity dependency among attributes, while previous works loose insight of this.

7 Conclusion

In this paper, we studied the duplicate identification problem in the context of deep web data integration. The proposed solution can build one universal matcher for multiple web databases in one domain instead of ones. We believe this is the first try to address the duplicate identification problem by building one universal matcher. Our solution shows better performance when web databases have little spell errors and have rich schemas.

Acknowledgement

This work was supported in part by the China Postdoctoral Science Foundation funded project under grant 20080440256 and 200902014, NSFC (60833005 and 60875033), National High-tech R&D Program (2009AA011904 and 2008AA01Z421), the Doctoral Fund of Ministry of Education of China (200800020002), and National Development and Reform Commission High-tech Program of China (2008-2441). The authors would also like to express their gratitude to the anonymous reviewers for providing some very helpful suggestions.

References

1. Arasu, A., Ganti, V., Kaushik, R.: Efficient exact set-similarity joins. In: VLDB 2006 (2006)
2. Bilenko, M., Mooney, R.J., Cohen, W.W.: Adaptive Name Matching in Information Integration. IEEE Intelligent Systems 18(5) (2003)
3. Bayardo, R.J., Ma, Y.: Scaling up all pairs similarity search. In: WWW 2007 (2007)

4. Cohen, W.W.: Data Integration Using Similarity Joins and a Word-Based Information Representation Language. ACM Trans. Information Systems (3) (2000)
5. Chaudhuri, S., Chen, B., Ganti, V.: Example-driven design of efficient record matching queries. In: VLDB 2007 (2007)
6. Chang, K.C., He, B., Li, C., Patel, M., Zhang, Z.: Structured Databases on the web: Observations and Implications. SIGMOD Record 33(3) (2004)
7. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Trans. Knowl. Data Eng. 19(1) (2007)
8. Galhardas, H., Florescu, D., Shasha, D., Simon, E., Saita, C.: Declarative Data Cleaning: Language, Model, and Algorithms. In: VLDB 2001 (2001)
9. Koudas, N., Sarawagi, S., Srivastava, D.: Record linkage: similarity measures and algorithms. In: SIGMOD 2006 (2006)
10. Poon, H., Domingos, P.: Joint inference in information extraction. In: AAAI 2007 (2007)
11. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2) (2006)
12. `http://www.cs.utexas.edu/users/ml/riddle/index.html`
13. Shen, W., DeRose, P., Vu, L.: Source-aware Entity Matching: A Compositional Approach. In: ICDE 2007 (2007)
14. Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. In: SIGMOD (2004)
15. Smith, T.-F., Waterman, M.-S.: Identification of common molecular subsequences. Journal of Molecular Biology (1981)
16. Winkler, W.E.: Methods for Record Linkage and Bayesian Networks. Technical Report Statistical Research Report Series RRS2002/05, US Bureau of the Census (2002)
17. Winkler, W.E.: The state of record linkage and current research problems. US Bureau of Census (1999)
18. Xiao, C., Wang, W., Lin, X.: Efficient similarity joins for near duplicate detection. In: WWW 2008 (2008)