International Journal of Computer and Communication Technology

Volume 7 | Issue 4 Article 5

October 2016

RECORD MATCHING FOR WEB DATABASES BY DOMAIN-SPECIFIC QUERY PROBING

V. ALEKHYA
CSE Department, Vignan university, ale.vemulapalli1@gmail.com

DS.BHUPAL NAIK
CSE Department, Vignan university, dsbhupal@gmail.com

RECORD MATCHING FOR WEB DATABASES BY DOMAIN-SPECIFIC QUERY PROBING

V.ALEKHYA[1] & DS.BHUPAL NAIK[2]

[1,2]CSE Department, Vignan university
E-mail : ale.vemulapalli1@gmail.com, dsbhupal@gmail.com

Abstract - Record matching refers to the task of finding entries that refer to the same entity in two or more files, is a vital process in data integration. Most of the record matching methods are supervised, which requires the user to provide training data. These methods are not applicable for web database scenario, where query results dynamically generated on-the- fly. To address the problem of record matching in the Web database scenario, we present an unsupervised, online record matching method, UDD, which effectively identifies the duplicates from query result records of multiple web databases. First, same source duplicates are eliminated by using exact matching method the “presumed” non duplicate records from the same source can be used as training examples . Starting from the non duplicate set, we use two cooperating classifiers a weight component similarity summing classifier and an SVM classifier, to iteratively identify duplicates in the query results from multiple Web databases.

Keywords - *Record Matching, duplicate detection, record linkage, data deduplication, SVM.*

I. INTRODUCTION

Today, more and more databases that dynamically generate Web pages in response to user queries are available on the Web. These Web databases compose the deep or hidden Web, which is estimated to contain a much larger amount of high quality, usually structured information and to have a faster growth rate than the static Web. Most Web databases are only accessible via a query interface through which users can submit queries. Once a query is received, the Web server will retrieve the corresponding results from the back-end database and return them to the user. To build a system that helps users integrate and, more importantly, compare the query results returned from multiple Web databases, a crucial task is to match the different sources’ records that refer to the same real-world entity. The problem of identifying duplicates, that is, two (or more) records describing the same entity, has attracted much attention from many research fields, including Databases, Data Mining, Artificial Intelligence, and Natural Language Processing. Most previous work is based on predefined matching rules hand-coded by domain experts or matching rules learned offline by some learning method from a set of training examples. Such approaches work well in a traditional database environment, where all instances of the target databases can be readily accessed, as long as a set of high-quality representative records can be examined by experts or selected for the user to label.. Consequently, hand-coding or offline-learning approaches are not appropriate in web database scenarios for two reasons. First, the full data set is not available beforehand, and therefore, good representative data for training are hard to obtain. Second, and most importantly, even if good representative data are found and labeled for learning, the rules learned on the representatives of a full data set may not work well on a partial and biased part of that data set. problem Definition Our focus is on Web databases from the same domain, i.e., Web databases that provide the same type of records in response to user queries.

II. BACKGROUND WORK

Most record matching methods adopt a framework that uses two major steps([5]&[6]):

1. Identifying a similarity function : Using training examples (i.e., manually labeled duplicate and non duplicate records) and a set of predefined basis similarity measures/functions over numeric and/or string fields, a single composite similarity function over one pair of records, which is a weighted combination (often linear) of the basis functions, is identified by domain experts or learned by a learning method, such as Expectation-Maximization, decision tree, Bayesian network, or SVM ([1],[2],[3]&[4])

2. Matching records : The composite similarity function is used to calculate the similarity between the candidate pairs and highly similar pairs are matched and identified as referring to the same entity.

Problem Definition*:* Suppose there are s records in data source A and there are t records in data source B, with each record having a set of fields/attributes. Each of the t records in data source B can potentially be a duplicate of each of the s records in data source A. The goal of duplicate detection is to determine the matching status, i.e., duplicate or non duplicate, of these s Â t record pairs.

An important aspect of duplicate detection is to reduce the number of record pair comparisons. Several methods have been proposed for this purpose including standard blocking[9] sorted neighborhood method Bigram Indexing, and record clustering[1] . Even though these methods differ in how to partition the data set into blocks, they all considerably reduce the number of comparisons by only comparing records from the same block. Since any of these methods can be incorporated into UDD to reduce the number of record pair comparisons, we do not further consider this issue. While most previous record matching work is targeted at matching a single type of record, more recent work has addressed the matching of multiple types of records with rich associations \between the records. Even though the matching complexity increases rapidly with the number of record types, these works manage to capture the matching dependencies between multiple record types and utilize such dependencies to improve the matching accuracy of each single record type. Unfortunately, however, the dependencies among multiple record types are not available for many domains. Compared to these previous works, UDD is specifically designed for the Web database scenario where the records to match are of a single type with multiple string fields. These records are heavily query-dependent and are only a partial and biased portion of the entire data, which makes the existing work based on offline learning inappropriate. Moreover, our work focuses on studying and addressing the field weight assignment issue rather than on the similarity measure. In UDD, any similarity measure, or some combination of them, can be easily incorporated.

III. UDD METHOD

To overcome such problems, we propose a new record matching method Unsupervised Duplicate Detection (UDD) for the specific record matching problem of identifying duplicates among records in query results from multiple Web databases.

The key ideas of our method are: We focus on techniques for adjusting the weights of the record fields in calculating the similarity between two records.

1. Two records are considered as duplicates if they are "similar enough" on their fields. As illustrated by the previous example, we believe different fields may need to be assigned different importance weights in an adaptive and dynamic manner.
2. Due to the absence of labeled training examples, we use a sample of universal data consisting of record pairs from different data sources as an approximation for a negative training set as well as the record pairs from the same data source. We believe, and our experimental results verify, that doing so is reason- able since the proportion of duplicate records in the universal set is usually much smaller than the proportion of non duplicates.

Employing two classifiers that collaborate in an iterative manner, UDD identifies duplicates as follows: First, each field's weight is set according to its "relative distance," i.e., dissimilarity, among records from the approximated negative training set. Then, the first classifier, which utilizes the weights set in the first step, is used to match records from different data sources. Next, with the matched records being a positive set and the non duplicate records in the negative set, the second classifier further identifies new duplicates. Finally, all the identified duplicates and non- duplicates are used to adjust the field weights set in the first step and a new iteration begins by again employing the first classifier to identify new duplicates. The iteration stops when no new duplicates can be identified.

C1 —Weighted Component Similarity Summing (WCSS) Classifier

In our algorithm, classifier C1 plays a vital role. At the beginning, it is used to identify some duplicate vectors when there are no positive examples available. Then, after iteration begins, it is used again to cooperate with C2 to identify new duplicate vectors. Because no duplicate vectors are available initially, classifiers that need class information to train, such as decision tree and NaıveBayes, cannot be used. An intuitive method to identify duplicate vectors is to assume that two records are duplicates if most of their fields that are under consideration are similar. On the other hand, if all corresponding fields of the two records are dissimilar, it is unlikely that the two records are duplicates. To evaluate the similarity between two records, we combine the values of each component in the similarity vector for the two records. Different fields may have different importance when we decide whether two records are duplicates. The importance is usually data-dependent, which, in turn, depends on the query in the Web database scenario.

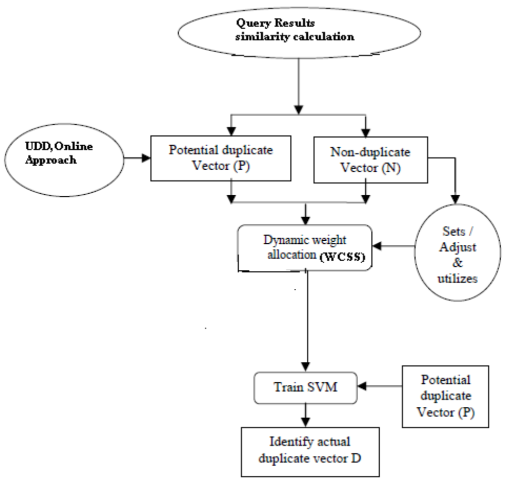
Hence, we define the similarity between records r1 and r2

$$Sim(r_1,r_2) = \sum_{i=1}^{n} w_i \cdot v_i,$$

where

$$\sum_{i=1}^{n} w_i = 1$$

System Architecture


Query Results
similarity calculation
UDD, Online Approach
Potential duplicate Vector (P)
Non-duplicate Vector (N)
Dynamic weight allocation (WCSS)
Sets / Adjust & utilizes
Train SVM
Potential duplicate Vector (P)
Identify actual duplicate vector D

and wi €[0,1] is the weight for the ith similarity component which represents the importance of the ith field. The similarity Sim(r1, r2) between records r1 and r2 will be in [0,1] according to the above definition..Duplicate Identification After we assign a weight for each component, the duplicate vector detection is rather intuitive. Two records r1 and r2 are duplicates if Sim(r1 , r2) Tsim , i.e., if their similarity value is equal to or greater than a similarity threshold. In general, the similarity threshold Tsim should be close to 1 to ensure that the identified duplicates are correct. Increasing the value of Tsim will reduce the number of duplicate vectors identified by C1 while, at the same time, the identified duplicates will be more precise. C2 —Support Vector Machine Classifier After detecting a few duplicate vectors whose similarity scores are bigger than the threshold using the WCSS classifier, we have positive examples, the identified duplicate vectors in D, and negative examples, namely, the remaining no duplicate vectors in N 0 . Hence, we can train another classifier C2 and use this trained classifier to identify new duplicate vectors from the remaining potential duplicate vectors in P and the no duplicate vectors in N . A classifier suitable for the task should have the following characteristics. First, it should not be sensitive to the relative size of the positive and negative examples because the size of the negative examples is usually much bigger than the size of the positive examples. This is especially the case at the beginning of the duplicate vector detection iterations when a limited number of duplicates are detected. Another requirement is that the classifier should work well given limited training examples. Because our algorithm identifies duplicate vectors in an iterative way, any incorrect identification due to noise during the first several iterations, when the number of positive examples is limited, will greatly affect the final result.

Evaluation Metric : As in many other duplicate detection approaches, we report the overall performance using recall and precision, which are defined as follows:

$$precision = \frac{\#of\ Correctly\ Identified\ Duplicate\ Pairs}{\#of\ All\ Identified\ Duplicate\ Pairs}$$

$$recall = \frac{\#of\ Correctly\ Identified\ Duplicate\ Pairs}{\#of\ True\ Duplicate\ Pairs},$$

However, as indicated in [10], due to the usually imbalanced distribution of matches and non matches in the weight vector set, these commonly used accuracy measures are not very suitable for assessing the quality of record matching. The large number of no matches usually dominates the accuracy measure and yields results that are too optimistic. Thus, we also use the F-measure, which is the harmonic mean of precision and recall, to evaluate the classification quality [2]:

$$F - measure = \frac{2 \cdot precision \cdot recall}{(precision + recall)}.$$

IV. EXPERIMENTAL RESULTS

We ran UDD on the Cora data set and its three subsets individually when the similarity threshold Tsim =0:85.Although Cora is a noisy data set, our algorithm still performs well over it. UDD has a precision of 0.896, recall of 0.950, and F-measure of 0.923 over the Cora data set. We compared our results with other works that use all or part of the Cora data set. Bilenko and Mooney [1], in which a subset of the Cora data set is used, report an F-measure of 0.867. Cohen and Richman [2] report 0.99/0.925 for precision/recall using a subset of the Cora data set. Culotta and McCallum [7] report an F-measure of 0.908 using the full Cora data set. From this comparison, it can be seen that the performance of UDD is comparable to these methods, all of which require training examples.

Effect of the threshold Tsim. The iteration row in the below table indicates the number of iterations required for UDD to stop. It can be seen that the duplicate vector detection iterations stop very quickly. All of them stop by the fifth iteration. On the one hand, the smaller Tsim is, the more iterations are required and the higher the recall. This is because the WCSS classifier with smaller Tsim identifies more duplicates and most of them are correct duplicates. Hence, with more correct positive examples, the SVM classifier can also identify more duplicates, which, in turn, results in a higher recall. On the other hand, the smaller Tsim is, the lower the precision. This is because the WCSS classifier with smaller Tsim is more likely to identify incorrect duplicates, which may incorrectly guide the SVM classifier to identify new incorrect duplicates. In our experiments, the highest F- measures were achieved when Tsim =0:85 over all the four data sets Performance of UDD on the Web Database Data Sets

	Precision	Recall	F-measure	Avg. Execution Time (sec)
Book-full	0.954	0.925	0.939	0.85
Book-titau	0.947	0.952	0.950	0.36
Hotel	0.961	0.952	0.955	0.74
Movie	0.932	0.928	0.930	0.21

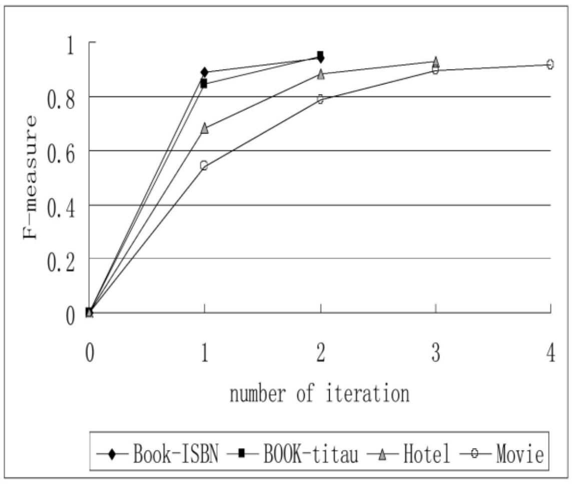


Fig. 1 : Performance of UDD at the end of each Duplicate detection iteration over the four Datasets when T_{sim}=0.85

	T_{sim}	0.75	0.8	0.85	0.9	0.95
Book-full	iteration	4	4	2	2	2
	precision	.767	.80	.954	.955	.959
	recall	.964	.935	.925	.908	.874
	F-measure	.854	.862	.939	.931	.914
Book-titau	iteration	4	3	2	2	2
	precision	.755	.806	.954	.961	.969
	recall	.951	.947	.948	.908	.854
	F-measure	.842	.871	.951	.934	.908
Hotel	iteration	5	4	3	2	2
	precision	.732	.865	.956	.962	.963
	recall	.987	.965	.950	.902	.807
	F-measure	.841	.912	.953	.931	.878
Movie	iteration	5	5	4	3	2
	precision	.717	.817	.943	.984	.986
	recall	.965	.942	.921	.85	.80
	F-measure	.823	.875	.932	.912	.882

Fig. 2 : Performance of UDD with Different Tsim on the Web Database Data Sets

Effect of the number of iterations:

The above figure 1 shows the performance of UDD at the end of each duplicate detection iteration over the four Web database data sets when Tsim = 0:85. It can be seen that the iteration stops quickly for all data sets and takes at most four iterations.

Figure 2 shows UDD's performance when using five different similarity thresholds (Tsim: 0.75, 0.80, 0.85, 0.90,and 0.95) on the four Web database data sets. The iteration row in this table indicates the number of iterations required for UDD to stop. It can be seen that the duplicate vector detection iterations stop very quickly. All of them stop by the fifth iteration. On the one hand, the smaller Tsim is, the more iterations are required and the higher the recall. This is because the WCSS classifier with smaller Tsim identifies more duplicates and most of them are correct duplicates. Hence, with more correct positive examples, the SVM classifier can also identify more duplicates, which, in turn, results in a higher recall. On the other hand, the experiments, the highest F-measures were achieved when Tsim = 0:85 over all the four data sets. smaller Tsim is, the lower the precision. This is because the WCSS classifier with smaller Tsim is more likely to identify incorrect duplicates, which may incorrectly guide the SVM classifier to identify new incorrect duplicates. In our smaller compared with other fields. Thus, they gain larger weights. In turn, the vectors staying in N are even more unlikely to be identified as duplicates in the next iteration because of the larger weights on their fields with small similarity values. Consequently, a high Tsim value makes it difficult for the WCSS classifier to find new positive instances after the first two iterations.

We also observe from figure 2 that the iterations stop more quickly when the threshold Tsim is high. When Tsim is high, vectors are required to have more large similarity values on their fields in order to be identified as duplicates in the early iterations. Hence, vectors with only a certain number of fields having large similarity values and other fields having small similarity values are likely to stay in the negative example set N. Recall that, according to the nonduplicate intuition, when setting the component weights in the WCSS classifier, fields with more large similarity values in N gain smaller weights and fields with more small similarity values in N gain larger weights. The vectors that stay in N would make the fields with small similarity values on all vectors in N relatively even smaller compared with other fields. Thus, they gain larger weights. In turn, the vectors staying in N are even more unlikely to be identified as duplicates in the next iteration because of the larger weights on their fields with small similarity values. Consequently, a high Tsim value makes it difficult for the WCSS classifier to find new positive instances after the first two iterations.

Influence of duplicate records from the same data source. Recall that, in this Section , we assumed that records from the same data source are non duplicates so that we can put pairs of them into the negative example set N. However, we also pointed out that, in reality, some of these record pairs could be actual duplicates. We call the actual duplicate vectors in N false non duplicate vectors. The number of false non duplicate vectors increases as the duplicate ratio,defined in Definition 1 in this Section, increases.The false nonduplicate vectors will affect the two classifiers in our algorithm in the following ways:

1. While setting the component weights W in WCSS according to the nonduplicate intuition, components that usually have large similarity values in N will be assigned small weights. Note that the false nonduplicate vectors are actually duplicate vectors with large similarity values for their components. As a result, an inappropriate set of weights could be set for WCSS.
2. Since the SVM classifier learns by creating a hyperplane between positive and negative examples, the false nonduplicate vectors will incorrectly add positive examples in the negative space. As a result,the hyperplane could be moved toward the positive space and the number of identified duplicate vectors will be much smaller.

V. CONCLUSION

Duplicate detection is an important step in data integration and most state-of-the-art methods are based on offline learning techniques, which require training data. In the Web database scenario, where records to match are greatly query-dependent, a pertained approach is not applicable as the set of records in each query's results is a biased subset of the full data set. To overcome this problem, we presented an unsupervised, online approach, UDD, for detecting duplicates over the query results of multiple Web databases. Two classifiers, WCSS and SVM, are used cooperatively in the convergence step [10] of record matching to identify the duplicate pairs from all potential duplicate pairs iteratively. Experimental results show that our approach is comparable to previous work that requires training examples for identifying duplicates from the query results of multiple Web databases

VI. ACKNOWLEDGEMENTS

This research was supported by the vignan university, guntur. Iam thankful to our guide who helpful for doing my research

REFERENCES

[1] M. Bilenko and R.J. Mooney, "Adaptive Duplicate DetectionUsing Learnable String Similarity Measures," Proc.ACMSIGKDD,pp.39-48,2003.

[2] W.W. Cohen and J. Richman, "Learning to Match and Cluster Large High-Dimensional Datasets for Data Integration," Proc. ACM SIGKDD,pp.475-480,2002

[3] Y. Thibaudeau, "The Discrimination Power of DependencyStructures in Record Linkage," Survey Methodology, vol. 19,pp. 31-38, 1993.

[4] W.E. Winkler, "Using the EM Algorithm for Weight Computation in the Fellegi-Sunter Model of Record Linkage," Proc. Section Survey Research Methods, pp. 667-671, 1988.

[5] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios, "Duplicate Record Detection: A Survey," IEEE Trans. Knowledge and Data Eng.,vol. 19, no. 1, pp. 1-16, Jan. 2007.

[6] N. Koudas, S. Sarawagi, and D. Srivastava, "Record Linkage:Similarity Measures and Algorithms (Tutorial)," Proc. ACMSIGMOD, pp. 802-803, 2006.

[7] A. Culotta and A. McCallum, "A Conditional Model of Deduplication for Multi-Type Relational Data," Technical Report IR-443, Dept. of Computer Science, Univ. of Massachusetts Amherst

[8] M.A. Hernandez and S.J. Stolfo, "The merge/Purge Problem for Large Databases," ACM SIGMOD Record, vol. 24, no. 2, pp. 127-138, 1995.

[9] M.A. Jaro, "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida," J. Am. Statistical Assoc., vol. 89, no. 406, pp. 414-420, 1989.