Multi-Domain Record Matching over Query Results from Multiple Web Databases

P.Kowsiga, T.Mohanraj

Abstract— Record Matching is a process to identify the duplicate records in web databases. It is an important step for data integration. In earlier systems, the record matching is addressed through the Unsupervised Online Record Matching method, UDD, i.e for a given user query, can effectively identify duplicates from the query result records of multiple web databases. This process of record matching are done through a single domain which provides limited number of non-duplicate data results. Hence, the proposal is made for a Multi-domain record matching process which includes an algorithm called N-Staged SVM, that helps to separate the duplicate and non-duplicate records based on the classifiers. The N-Staged SVM which helps to separate the duplicate and non-duplicate data using iterative process. A single domain can include multiple web databases, a single database can include multiple hyperplanes, a single hyperplane include multiple data, which are made separated as duplicate and non-duplicate using the N-Staged SVM. This process is repeated for multiple domains by constructing hyperplanes for each. Hence the result produced will be efficient and more reliable results are provided for the user query.

Index Terms—Record Matching, UDD, Duplicates, Multi-Domain, SVM, N-Staged SVM, Hyperplanes.

1 INTRODUCTION

Now-a-days, numerous number of databases that generate web pages in accordance to the user queries on the web. These web databases includes the data that are redundant, unreliable etc., To remove/ignore the duplicates/redundant data an unsupervised duplicate detection (UDD) and SVM are made used. This method can remove duplicates for multiple web databases, where a single domain can include multiple web databases, similarly the user query can include multiple domains. So a system is build that helps the users to compare query result returned from multiple domain, multiple databases, i.e., to match the different sources records that refer to the same real-world entity, to match records that are identical.

The problem of identifying duplicates, i.e., identical records, where in previous work is based on the WCSS and SVM, i.e., based on the weight of the data, if two or more data having same weight are identified as duplicate and SVM are used to separate the duplicate and non-duplicate. These are possible only for multiple web databases of single domain and provides limited results.

A new approach called N-Staged SVM are made used which checks for duplicates in multiple domain simultaneously and provides enormous results which are free from duplicates. The general SVM classifier, classifies the duplicate and non-duplicate data from the web databases for single domain.

The N-Staged SVM which checks for a part of two data sets and separates the duplicate and non-duplicate, the resulted non-duplicate used as the input for other comparison of data. This is how the N-Staged SVM are made used in the multi-domain Record matching for user query results.

To illustrate this problem, the proposed project have taken a paper for study which contains the example as follows, consider a query for books of a specific author, such as "J. K. Rowling." Depending on how the Web databases process such a query, all the result records for this query may well have only "J. K. Rowling" as the value for the Author field. In this case, the Author field of these records is ineffective for distinguishing the records that should be matched and those that should not.

To overcome these problems, an Unsupervised Duplicate Detection (UDD) for the specific record matching problem of identifying duplicates among records in query results from multiple Web databases. UDD focused on techniques for adjusting the weights of the record fields in calculating the similarity between two records. Two records are considered as duplicates if they are "similar enough" on their fields.

2 DUPLICATE DETECTION IN UDD

2.1 PROBLEM DEFINITION

The 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. 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 nonduplicate, of these s _ t record pairs.

2.1.1 DUPLICATE DEFINITION

Different users may have different criteria for what constitutes

a duplicate even for records within the same domain. For example, if the user is only interested in the title and author of a book and does not care about the ISBN information, Furthermore, the records numbered and are also duplicates under this criterion. In contrast, some users may be concerned about the ISBN field besides the title and author fields. For these users, the records numbered and the second record are not duplicates. This user preference problem makes supervised duplicate detection methods fail. Since UDD is unsupervised, it does not suffer from this problem.

2.1.2 PROBLEM FORMULATION

Initially, two sets of vectors can be built.

1. A nonduplicate vector set N that includes similarity vectors formed by any two different records from the same data source.
2. A potential duplicate vector set P that includes all similarity vectors formed by any two records from different data sources.

2.2 UDD ALGORITHM OVERVIEW

Input: Potential duplicate vector set *P*
Non-duplicate vector set *N*
Output: Duplicate vector set *D*
C_1: a classification algorithm with adjustable parameters *W* that identifies duplicate vector pairs from *P*
C_2: a supervised classifier, e.g., SVM

Algorithm:
1. D = Ø
2. Set the parameters *W* of C_1 according to *N*
3. Use C_1 to get a set of duplicate vector pairs d_1 from *P*
4. Use C_1 to get a set of duplicate vector pairs f from *N*
5. $P = P - d_1$
6. While $|d_1| \neq 0$
7. $N' = N - f$
8. $D = D + d_1 + f$
9. Train C_2 using *D* and *N'*
10. Classify *P* using C_2 and get a set of newly identified duplicate vector pairs d_2
11. $P = P - d_2$
12. $D = D + d_2$
13. Adjust the parameters *W* of C_1 according to *N'* and *D*
14. Use C_1 to get a new set of duplicate vector pairs d_1 from *P*
15. Use C_1 to get a new set of duplicate vector pairs f from *N*
16. *N*=*N'*
17. Return *D*

Fig.2. Duplicate vector identification algorithm

The overall UDD algorithm is presented in Fig. 2.

An intuitive solution to this problem is that we can learn a classifier from N and use the learned classifier to classify P. Although there are several works based on learning from only positive (or negative) examples, to our knowledge all works in the literature assume that the positive (or negative) examples are all correct. However, N may contain a small set of false negative examples. **Weighted Component Similarity Summing Classifier**

2.3 C_1—WEIGHTED COMPONENT SIMILARITY SUMMING CLASSIFIER

In WCSS algorithm, classifier C_1 plays 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 C_2 to identify new duplicate vectors. 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. Hence, we define the similarity between records r_1 and r_2 as

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

Where,

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

and $w_i \in [0,1]$ is the weight for the i^{th} similarity component, which represents the importance of the i^{th} field. The similarity $Sim(r_1, r_2)$ between records r_1 and r_2 will be in [0,1] according to the above definition. Because no duplicate vectors are available initially, classifiers that need class information to train, such as decision tree and Naıve Bayes, cannot be used. On the other hand, if all corresponding fields of the two records are dissimilar, it is unlikely that the two records are duplicates.

2.3.1 COMPONENT WEIGHT ASSIGNMENT

In the WCSS classifier, we assign a weight to a component to indicate the importance of its corresponding field under the condition that the sum of all component weights is equal to 1. The intuition for the weight assignment includes:

1. *Duplicate intuition*: The similarity between two duplicate records should be close to 1. For a duplicate vector V_{12} that is formed by a pair of duplicate records r_1 and r_2, we need to assign large weights to the components with large similarity values and small weights to the components with small similarity values.
2. *Nonduplicate intuition:* The similarity for two non duplicate records should be close to 0. Hence, for a non duplicate vector V_{12} that is formed by a pair of nonduplicate records r_1 and r_2, we need to assign small weights to the components with large similarity val-

ues and large weights to the components with small similarity values.

According to the duplicate intuition, the following weight assignment scheme designed considering all duplicate vectors in D:

$$P_i = \sum_{v \varepsilon D} v_i \longrightarrow (2)$$

and

$$w_{di} = \frac{P_i}{\sum_{j=1}^{n} P_j} \longrightarrow (3)$$

in which p_i is the accumulated ith component similarity value for all duplicate vectors in D and wdi is the normalized weight for the ith component. For each component, if it usually has a large similarity value in the duplicate vectors, pi will be large according to (2) and, in turn, a large weight will be assigned for the ith component according to (3). On the other hand, the component will be assigned a small weight if it usually has a small similarity value in the duplicate vectors. According to the nonduplicate intuition, we use the following weight assignment scheme considering all nonduplicate vectors in N:

$$q_i = \sum_{v \varepsilon N} (1 - v_i) \longrightarrow (4)$$

and

$$w_{ni} = \frac{q_i}{\sum_{j=1}^{n} q_j} \longrightarrow (5)$$

in which q_i is the accumulated i_{th} component dissimilarity value for all nonduplicate vectors in N and w_{ni} is the normalized weight for the i_{th} component. In a similarity vector $V = \langle v_1; v_2; \ldots; v_n \rangle$, the i^{th} component dissimilarity refers to $1-v_i$. For each component, if it usually has a large similarity value in the nonduplicate vectors, it will have a small accumulated dissimilarity according to (4) and will, in turn, be assigned a small weight according to (5). In our experiment, we give each scheme a weight to show its importance:

$$w_i = a.w_{di} + (1-a)w_{ni} \rightarrow (6)$$

in which $a \in [0,1]$ denotes the importance of duplicate vectors versus nonduplicate vectors. At the start of our algorithm in Fig. 2, there is no duplicate vector available. Hence, a is assigned to be 0. As more duplicate vectors are discovered, we increase the value of a. We initially set a to be 0.5 at the 2nd iteration to indicate that D and N are equally important and incrementally add 0.1 for each of the subsequent iterations.

2.3.1 DUPLICATE IDENTIFICATION

After we assign a weight for each component, the duplicate vector detection is rather intuitive. Two records r_1 and r_2 are duplicates if $Sim(r_1; r_2) \geq T_{sim}$, 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 T_{sim} will reduce the number of duplicate vectors identified by C_1 while, at the same time, the identified duplicates will be more precise. 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.

2.4 C_2—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 nonduplicate vectors in N′. 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 nonduplicate 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 will greatly affect the final result. Support Vector Machine (SVM), which is known to be insensitive to the number of training examples, satisfies all the desired requirements and is selected for use in UDD. Because algorithm will be used for online duplicate detection. 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. Support Vector Machine (SVM), which is known to be insensitive to the number of training examples, when the number of positive examples is limited, will greatly affect the final result. Support Vector Machine (SVM), which is known to be insensitive to the number of training examples, satisfies all the desired requirements and is selected for use in UDD.

3 N-STAGED SVM CLASSIFIER

Although, the UDD solves the problem of Duplicate detection, it just partially accomplished the task. The resulted query

results are limited as it extracted data from a single domain. This leads to the presence of duplicate data at the other domains which are not searched by the existing system.

To my knowledge, this is the first work that studies and solves the online duplicate detection problem for the web database for multi-domain scenario where query results are generated on the fly. To identify duplicate records in multi-domain, a classification algorithm is proposed, i.e., N-Staged SVM classifier.

SVM which constructs a space / hyper plane in a high /infinite dimensional space used for classification. In this project, problem to be dealt is to find duplicate records provided by user queries in multiple domains. Hence an iterative process of SVM is carried out i.e., 'n' hyper planes are constructed in accordance to the query and duplicates identified. This process of activity is called as N-Staged classification, for which an algorithm is proposed. The overall UDD algorithm is presented below:

Input : Vector, DS (Data Sets)
Domain, D_m; Items $i_0,i_1,\ldots,i_n$
Hyperplane, H
Output : Non-Duplicate Vector, N
Parameters: C_1 & C_2 classifiers with Rules
$R_0,R_1,\ldots,R_n$
Algorithm:
(1) $DS=DS\{i_0,i_1,..i_n\}$
(2) $Dm=D_{m0},D_{m1},..,D_{mn}$
(3) $H=H_0,H_1,\ldots,H_n$
(4) while $D_{mo} < D_{mn}$
(5) Form Hyperplane, H
(6) while $H_0 < H_n$
(7) Get the vector DS
(8) Form rules R_0

$$\sum_{DS_0}^{DS_n}\left(\sum_{R_0}^{R_n}\right)$$

(9) Plot items $i_0,i_1,..,i_n$ on H
(10) Create classifier C_1 & C_2
C_1 → Duplicate Set, D
C_2 → Non-Duplicate Set, N
(11) Extract item $i_0,i_1,\ldots,i_n$ on N
$N = \{i_0,i_1,\ldots i_n\}$
(12) Output N
(13) Loop (6)
(14) Loop (4)

Fig.3. N-Staged SVM algorithm

In this algorithm, the data set items are made differentiated by constructing hyper plane which identifies the duplicate and non-duplicate data that are separated by margins. The item that lies near the margin are considered as the Support Vector. Initially the dataset includes the number of items as $i_0,i_1,\ldots i_n$. A single domain which includes multiple hyper planes. A single hyper plane include multiple data i.e., $i_o,i_1..i_n$. these data sets can be formed based on some rules.

$$\sum_{DS_0}^{DS_n}\left(\sum_{R_0}^{R_n}\right)$$

The items are plotted on the hyper plane based on therules. The classifiers C_1(Duplicate set, D) and C_2 (Non-Duplicate set, N) are used to identify the duplicate and non-duplicate pairs. This is possible by the condition of $H_1 < H_n$. Process is repeatd auntil the user query are made extracted without duplicates.

3.1 HOW ALGORITHM WORKS?

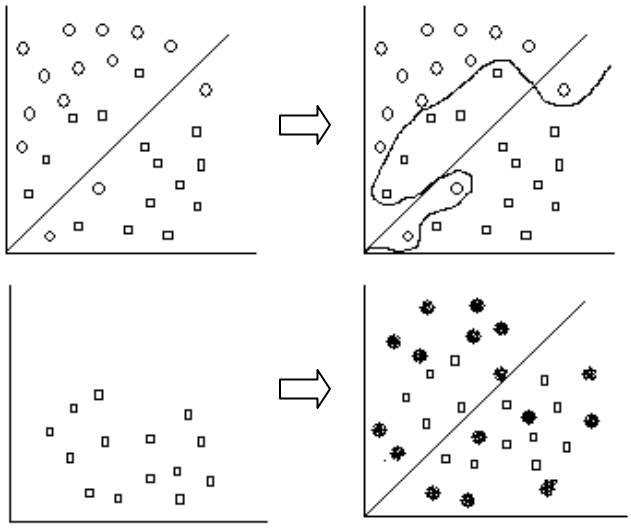

Fig.4. Algorithm Working

Where the upper left side hyperplane shows the two different itemsets which are made separated using a margin, which even has a mixing of items, i.e., the circle are one itemset and the squares are other itemset. Now, the upper right side shows a hyperplane which is separated using a Support Vector, so that the itemsets get separated and provides a result of non-duplicate datasets illustrated in lower left. Now this result used as input for next hyperplane shown in lower right. Hence this is how the SVM came into existence in this project, which separates two different itemsets, one as duplicate set and the rest as non-duplicate set.

4 BREAST CANCER DATASET

The multi-domain record matching can be tested in a dataset, one of the three domains provided by Oncology Institute that has appeared in machine learning literature, here the Breast cancer dataset are taken as an example, which includes some of the attributes and its information. Attributes 2 through 10 have been used to represent instances. Each instance has one of 2 possible classes: benign or malignant.

Attribute Information

#	Attribute	Domain
1.	Sample code number	id number
2.	Clump Thickness	1 - 10
3.	Uniformity of Cell Size	1 - 10
4.	Uniformity of Cell Shape	1 - 10
5.	Marginal Adhesion	1 - 10
6.	Single Epithelial Cell Size	1 - 10
7.	Bare Nuclei	1 - 10
8.	Bland Chromatin	1 - 10
9.	Normal Nucleoli	1 - 10
10.	Mitoses	1 – 10

Breast cancer represents the leading cause of cancer deaths in women today, and it is the most common type of cancer in women. The classification method is used here for breast cancer. They help detect breast cancer by dividing tumors into needed categories.

5 CONCLUSION

Duplicate detection is an important step in data integration to extract duplicate free information and the problem of duplicate detection is done through UDD which checks for duplicates in a single domain at a stretch, which provides less query result as search is done in single domain. Although, the UDD solves the problem of Duplicate detection, it just partially accomplished the task. The resulted query results are limited as it extracted data from a single domain.

To overcome this problem, a new proposal which includes some more features that can able to solve the existing system problems in ease. N-Staged SVM algorithm is proposed which provides more results by detecting the duplicate records at multiple domains simultaneously by separating each itemset in the domain through the SVM classification, similarly the process continues upto N-Stages until the duplicates are removed and the user query are extracted with non-duplicate data which are accurate result.

Hence by the above process, more number of results are gained by the search to multiple domains at a stretch and those results are reliable to the user and in this way N-Staged SVM overcomes the UDD algorithm.

ACKNOWLEDGMENT

Myself(P.KOWSIGA) wish to thank to my university, to my HOD (Mr.P.S.Balamurugan) and to my guide (Mr.T.Mohanraj) who has encouraged a lot. I extend my hearty thanks to my parents (Mr. K.Palanisamy and Mrs.P.Thulasimani) who has suppored a lot financially and for encouragement over publishing this journal .

REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, "Eliminating Fuzzy Duplicates in Data Warehouses," Proc. 28th Int'l Conf. Very Large Data Bases, pp. 586-597, 2002.

[2] M. Bilenko and R.J. Mooney, "Adaptive Duplicate Detection Using Learnable String Similarity Measures," Proc. ACM SIGKDD, pp. 39-48, 2003.

[3] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and Efficient Fuzzy Match for Online Data Cleaning," Proc. ACM SIGMOD, pp. 313-324, 2003.

[4] P.Christen, T.Churches, and M.Hegland, "Febrl-A Parallel Open Source Data Linkage System," Advances in Knowledge Discovery and Data Mining, pp. 638-647, Springer, 2004.

[5] S. Chaudhuri, V. Ganti, and R. Motwani, " Robust Identification of Fuzzy Duplicates," Proc. 21st IEEE Int'l Conf. Data Eng., pp. 865-876, 2005.

[6] H.Zhao, W.Meng, A.Wu, V.Raghavan, and C.Yu, "Fully Automatic Wrapper Generation for Search Engines," Proc. 14th World Wide Web Conf., pp. 66-75, 2005.

[7] B. He and K.C.-C. Chang, "Automatic Complex Schema Matching Across Web Query Interfaces: A Correlation Mining Approach," ACM Trans. Database Systems, vol. 31, no. 1, pp. 346-396, 2006.

[8] Y. Zhai and B. Liu, "Structured Data Extraction from the Web Based on Partial Tree Alignment," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 12, pp. 1614-1628, Dec. 2006.

[9] 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.

[10] P. Christen, "Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification," Proc. ACM SIGKDD, pp. 151-159, 2008.

[11] W.Su, J.Wang, and F.H.Lochovsky, "Record Matching over Query Results from Multiple Web Databases," IEEE Trans.Knowledge and Data Eng., vol. 22, no. 4, pp. 578-589, 2010.

AUTHORS BIOGRAPHY

P.Kowsiga received her M.Sc (Software Engineering) Degree in the year 2010, from M.Kumarasamy College of Engineering, Karur - Affiliated to Anna University, Chennai. Currently she is pursuing M.E (Computer Science and Engineering) in Karpagam University, Coimbatore. Her Area of interest includes Data Mining.

T.Mohanraj received his M.Sc (Computer puter Science) Degree in the year 2006, from Bharathiar University, Coimbatore tore and received his M.E (Computer Science and Engineering) Degree in the year 2009, from Anna University, Coimbatore. Currently he is working as Asst.Professor in the Department of Computer Science and Engineering at Karpagam University, Coimbatore. His area of interest includes Web Mining.