ISBN 978-93-5156-328-0

Record Matching Over Query Result from Multiple Web Databases

Gokul K. Bodke, Meenal S. Khairnar, Amol D. Potgantwar
Department of Computer Science and Engineering, SIEM, Nashik, India

Article Info

Article history:
Received 2 January 2014
Received in revised form
10 January 2014
Accepted 20 January 2014
Available online 1 February 2014

Keywords

Record Matching,
Duplicate detection,
UDD

Abstract

Record matching, is the process of identifying the records that represent the same real-world entity, is an important step for data integration. Most record matching methods are supervised, which requires the user to provide training data. These methods are not applicable for the Web database scenario, where the records to match are query results dynamically generated, on the fly, such records are query-dependent and a pre-learned method using training examples from previous query results may fail on the results of a new query. To address the problem of record matching in the Web database scenario, we present an unsupervised, online record matching method, UDD, which, for a given query, can effectively identify duplicates from the query result records of multiple Web databases. This method has two cooperating classifiers, a Weighted Component Similarity Summing classifier (WCSS) and Support Vector Machine classifier (SVM), to iteratively identify duplicates in the query results from multiple Web databases. Using these two classifiers duplicate and non-duplicate vectors are calculated and non-duplicate vector is displayed as result.

1. Introduction

The end user has no control over the results returned by a search engine, nor can he guarantee that there will be no duplicates from the query result. The problem of duplicate records existing in a query result referring to the same real-world entity can occur when search engine uses multiple web databases. 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. 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. Now a day's 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.

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:

1. We focus 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. 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 sources5 as an approximation for a negative training set as well

Corresponding Author,
E-mail address:

as the record pairs from the same data source. We believe, and our experimental results verify, that doing so is reasonable 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.[1]

2. Literature Survey

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 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. While most previous record matching work is targeted at matching a single type of record. Unfortunately, however, the dependencies among multiple record types are not available for many domains. Some of the existing technologies for duplicate detection are given below:

2.1 Adaptive Duplicate Detection Using Learnable String Similarity Measures

The problem of identifying approximately duplicate records in databases is an essential step for data cleaning and data integration processes. Most existing approaches have relied on generic or manually tuned distance metrics for estimating the similarity of potential duplicates. In this paper, we present a framework for improving duplicate detection using trainable measures of textual similarity. We propose to employ learn able text distance functions for each database field, and show that such measures are capable of adapting to the specific notion of similarity that is appropriate for the field's domain. We present two learn able text similarity measures suitable for this task: an extended variant of learnable string edit distance, and a novel vector-space based measure that employs a Support Vector Machine (SVM) for training. Experimental results on a range of data sets show that our framework can improve duplicate detection accuracy over traditional techniques. Databases frequently contain field-values and records that refer to the same entity but are not syntactically identical. Variations in representation can arise from typographical errors, misspellings, abbreviations, as well as integration of multiple data sources. Such approximate duplicates can have many deleterious effects, including preventing data-mining algorithms from discovering important regularities. This problem is typically handled during a tedious manual data cleaning, or "de-duping", process.[5]

2.2 Swoosh: a generic approach to entity resolution

We consider the Entity Resolution (ER) problem (also known as reduplication, or merge-purge), in which records determined to represent the same real-world entity are successively located and merged. We formalize the generic ER problem, treating the functions for comparing and merging records as black-boxes, which permits expressive and extensible ER solutions. We identify four important properties that, if satisfied by the match and merge functions, enable much more efficient ER algorithms. We develop three efficient ER algorithms: G-Swoosh for the case where the four properties do not hold, and R-Swoosh and F-Swoosh that exploit the 4 properties. F-Swoosh in addition assumes knowledge of the "features" (e.g., attributes) used by the match function. We experimentally evaluate the algorithms using comparison shopping data from Yahoo! Shopping and hotel information data from Yahoo! Travel. We also show that R-Swoosh (and F-Swoosh) can be used even when the four match and merge properties do not hold, if an "approximate" result is acceptable. Entity Resolution (ER) (sometimes referred to as reduplication) is the process of identifying and merging records judged to represent the same real-world entity. ER is a well known problem that arises in many applications. [3]

2.3 Robust and Efficient Fuzzy Match for Online Data Cleaning

To ensure high data quality, data warehouses must validate and cleanse incoming data tuples from external sources. In many situations, clean tuples must match acceptable tuples in reference tables. For example, product name and description fields in a sales record from a distributor must match the pre-recorded name and description fields in a product reference relation. A significant challenge in such a scenario is to implement an efficient and accurate fuzzy match operation that can effectively clean an incoming tuple if it fails to match exactly with any tuple in the reference relation. In this paper, we propose a new similarity function which overcomes limitations of commonly used similarity functions, and develop an efficient fuzzy match algorithm. We demonstrate the effectiveness of our techniques by evaluating them on real datasets. [3]

2.4 Eliminating Fuzzy Duplicates in Data Warehouses

The duplicate elimination problem of detecting multiple tuples, which describe the same real world entity, is an important data cleaning problem. Previous domain independent solutions to this problem relied on standard textual similarity functions (e.g., edit distance, cosine metric) between multi-attribute tuples. However, such approaches result in large numbers of false positives if we want to identify domain-specific abbreviations and conventions. In this paper, we develop an algorithm for eliminating duplicates in dimensional tables in a data warehouse, which are usually associated with hierarchies. We exploit hierarchies to develop a high quality, scalable duplicate elimination algorithm, and evaluate it on real datasets from an operational data warehouse. Decision support analysis on data warehouses influences important business decisions; therefore, accuracy of such analysis is crucial. However, data received at the data warehouse from external sources usually contains errors: spelling mistakes, inconsistent conventions, etc. Hence, significant amount of time and money are spent on *data cleaning*, the task of detecting and correcting errors in data.[4]

3. System Architecture

System architecture is mainly divided into 3 parts:
1 .Google Crawler
2. DOM Parser
3. Duplicate Detection

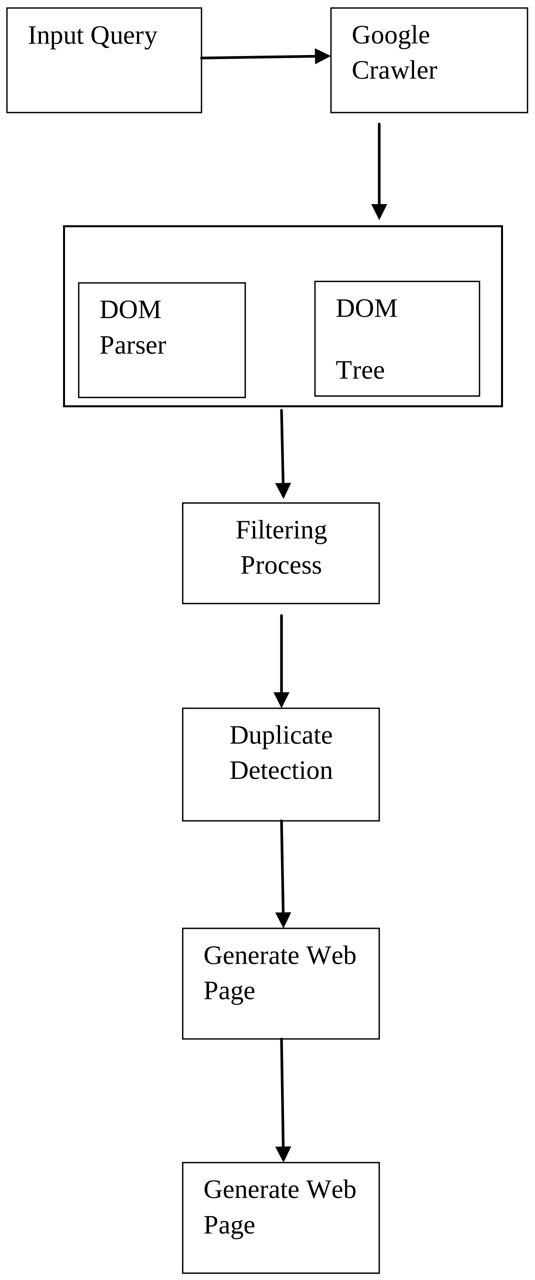

Fig: 1. System Architecture

Here, user will enter the query in the text field, in natural language. Then this query is send to google crawler. It will extract the data from network. crawler is nothing but a program that visits web sites and read their pages and other information to create entries for a search engine index. Then the result is pass to DOM parser, DOM is Document Object Parser, it is a cross platform or a language independent convention for representing and interacting with objects in HTML/XML. DOM parser will create a DOM tree.
E.g. consider a simple part of HTML code,

```
<table>
<tbody>
<tr>
<td> Green </td>
<td> Red </td>
</tr>
<tr>
<td> Yellow </td>
<td> White </td>
</tr>
```

The DOM tree for this part of code is created as follows

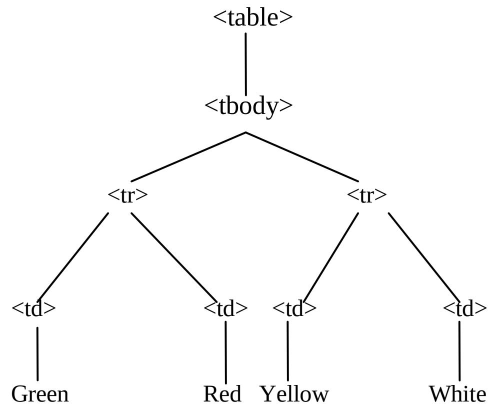

Fig: 3.2 DOM Tree

Then the main process of filtering is started, it will take only that leaf nodes which represents the *"href"* tag and all the other things are drop like image, data. After that duplicate detection [6] process is carried out, it uses two classifiers WCSS and SVM, these two classifiers create duplicate and non-duplicate vectors and then according to non-duplicate vector set it will generate a web page and display the result.

4. Methodology for Development

In the proposed system the plan is to develop an unsupervised algorithm that uses three classifiers for duplicate detection. The first classifier is called the Weighted Component Similarity Summing (WCSS) Classifier where the importance of the fields is determined and duplicates are identified. The idea for this classifier is to calculate the similarity between pair of records by doing a field to filed comparison. This serves as input to the second classifier which is the Support Vector Machine (SVM) Classifier which makes use of the duplicates and non-duplicates identified from the WCSS classifiers. The SVM classifier then processes each record to identify/classify a record as being a duplicate or otherwise. The major enhancement would be to add functions to the WCSS classifier which deal with string comparison that can aid in identifying duplicates. An ideal system would be one that presents the most relevant results from multiple web databases. This thesis was an attempt to explore using alternative methodologies for record duplicate detection. By developing additional classifiers, this thesis tried to prove that an unsupervised learning system is more suitable for duplicate detection with the flexibility of learning on the fly and adapting to the queries.

4.1 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. 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 as

$$Sim(r1,r2) = \sum_{i=1}^{n} wi.vi$$

Where

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

and wi belongs to 0; 1_ is the weight for the ith similarity component, 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.[1]

4.2 C2- Support Vector Machine (SVM) Classifier

After detecting a few duplicate vectors whose similarity scores are bigger than the threshold using the WCSS classifier, output of WCSS is applied as input to SVM classifier. SVM will again detect the duplicate vector pairs. The WCSS classifier outputs three sets of similarity vectors namely potential duplicate vectors, non duplicate vectors and identified duplicate vectors. From these vectors, the identified

ISBN 978-93-5156-328-0

duplicate vectors *D* are sent as positive examples and non-duplicate vectors *N* as negative examples for training purpose. These two vectors serve as input to the SVM classifier. We can train SVM classifier and use this trained classifier to identify new duplicate vectors from the potential duplicate vector *P*.[1]

5. Conclusion

This system concentrated on the development of an Unsupervised Duplicate Detection algorithm that can serve as foundation for developing applications that use Web databases. Two classifiers, WCSS and SVM, are used cooperatively in the convergence step of record matching to identify the duplicate pairs from all potential duplicate pairs iteratively. It detects the duplicate records from the results generated on the fly. It does not suffer from user preference problem. A faster better string similarity calculation is used for optimizing the performance of UDD.

References

[1] W. Su, J. Wang, F. H. Lochovsky, Record Matching over Query Results from Multiple Web Databases," IEEE Trans.Knowledge and Data Eng., 22(4), 578-589, 2010

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

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

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

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

[6] A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios, Duplicate Record Detection: A Survey, IEEE Trans. Knowledge and Data Eng., 19(1), 1-16, 2007

[7] M. A. Jaro, Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida, J. Am. Statistical Assoc., 89(406), 414-420, 1989

[8] N. Koudas, S. Sarawagi, D. Srivastava, Record Linkage: Similarity Measures and Algorithms (Tutorial), Proc. ACM SIGMOD, 802-803, 2006

[9] A. McCallum, K. Nigam, L. H. Ungar, Efficient Clustering of High-Dimensional Datasets with Application to Reference Matching, Proc. ACM SIGKDD, 169-178, 2000

[10]S H. Yu, J. Han, C. C. Chang, PEBL: Web Page Classification without Negative Examples, IEEE Trans. Knowledge and Data Eng., 16(1), 70-81, 2004