Duplicate Detection of Query Results from Multiple Web Databases

Hemalatha S
Department of Computer Science & Engineering
Adhiyamaan College of Engineering
Hosur - 635109, India

Raja K
Department of Computer Science & Engineering
Adhiyamaan College of Engineering
Hosur - 635109, India

Tholkappia Arasu
Department of CSE
Jayam college of Engineering and Technology,
Dharmapuri-636819, India

ABSTRACT
The results from multiple 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.The system that helps users integrate and, more importantly, compare the query results returned from multiple Web databases, an important task is to match the different sources' records that refer to the same real-world entity. Most state-of-the-art record matching methods are supervised, which requires the user to provide training data. In the Web databases, the records are not available in hand as they are query-dependent, they can be obtained only after the user submits the query. After removal of the same-source duplicates, the assumed non duplicate records from the same source can be used as training examples. The method uses the classifiers the weighted component similarity summing classifier (WCSS) and Support Vector Machine (SVM) classifier that works along with the Gaussian mixture model (GMM) to iteratively to identify the duplicates. The classifiers work cooperatively to identify the duplicate records. The complete GMM is parameterized by the mean vectors, covariance matrices and mixture weights from all the records.

General Terms
Data Mining, Record Matching, merge purge operation, Learning

Keywords
Gaussian Mixture Model, supervised learning, Support Vector Machine

1. INTRODUCTION
The Web Databases dynamically generate Web pages in response to user queries. Most Web databases are only accessible via a query interface through which users can submit queries. With many businesses, government organisations and research projects collecting massive amounts of data, the techniques collectively known as data mining have in recent years attracted interest both from academia and industry. While there is much ongoing research in data mining algorithms and techniques, it is well known that a large proportion of the time and effort in real-world data mining projects is spent understanding the data to be analysed, as well as in the data preparation and pre processing steps. An increasingly important task in the data pre processing step of many data mining projects is detecting and removing duplicate records that relate to the same entity within one data set. Similarly, linking or matching records relating to the same entity from several data sets is often required as information from multiple sources needs to be integrated, combined or linked in order to allow more detailed data analysis or mining. It takes advantage of the dissimilarity among records from the same Web database for record matching. Most existing work requires human-labeled training data (positive, negative, or both), which places a heavy burden on users. 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. In the Web database scenario, the records to match are highly query-dependent, since they can only be obtained through online queries. Moreover, they are only a partial and biased portion of all the data in the source Web databases.

To overcome such problems, propose a new method of record matching problem of identifying duplicates among records in query results from multiple web databases. The proposed approach Unsupervised Duplicate Detection[1] for the specific problem employs two classifiers that collaborate in an iterative manner. This paper proposes a method where the SVM classifier is fused with the Gaussian Mixture Model to identify the duplicates iteratively. The iteration stops when there are no more duplicates in the result set. Blocking methods [2] are used in record linkage systems to reduce the number of candidate record comparison pairs to a feasible number whilst still maintaining linkage accuracy. Blocking methods partition the data sets into blocks or clusters of records which share a blocking attribute or are otherwise similar with respect to a defined criterion.

2. RELATED WORK
The records consist of multiple fields, making the duplicate detection problem much more complicated. Approaches that rely on training data to learn how to match the records includes probabilistic approaches and supervised machine learning techniques. Approaches that rely on domain knowledge or on generic distance metrics to match records.

2.1 Probabilistic Matching Models
Let us assume that two classes M and U contains the record pairs that represent the same entity ("match") and the class U that contains the record pairs that represent two different entities ("nonmatch") [2],[3]. Let x be a comparison vector, randomly

drawn from the comparison space that corresponds to the record pair (a,b). The goal is to determine whether (a,b) ε M or (a,b) ε U. A decision rule(1), based simply on probabilities can be written as follows:

$$(a,b) \varepsilon \begin{cases} M & \text{if } p(M|x) \geq p(U|x) \\ U & \text{otherwise.} \end{cases} \quad (1)$$

2.2 Bigram Indexing

The Bigram Indexing (BI) method as implemented in the Febrl [4] record linkage system allows for fuzzy blocking. The basic idea is that the blocking key values are converted into a list of bigrams (sub-strings containing two characters) and sub-lists of all possible permutations will be built using a threshold (between 0.0 and 1.0). The resulting bigram lists are sorted and inserted into an inverted index, which will be used to retrieve the corresponding record numbers in a block. The number of sub-lists created for a blocking key value both depends on the length of the value and the threshold. The lower the threshold the shorter the sub-lists, but also the more sub-lists there will be per blocking key value, resulting in more (smaller blocks) in the inverted index. In the information retrieval field, bigram indexing has been found to be robust to small typographical errors in documents

2.3 Supervised and Semisupervised Learning

The supervised learning systems rely on the existence of training data in the form of record pairs, prelabeled as matching or not. One set of supervised learning techniques treat each record pair (a,b) independently, similar to the probabilistic techniques. A well-known CART algorithm [5], which generates classification and regression trees. A linear discriminant algorithm, which generates a linear combination of the parameters for separating the data according to their classes, and a "vector quantization" approach which is a generalization of the nearest neighbour algorithms. The transitivity assumption can sometimes result in inconsistent decisions. For example (a,b) and (a,c) can be considered matches, but (b,c) not. Partitioning such "inconsistent" graphs with the goal of minimizing inconsistencies in an NP-complete problem.

2.4 Active Learning Based Techniques

One of the problems with the supervised learning techniques is the requirement for a large number of training examples. While it is easy to create a large number of training pairs that are either clearly nonduplicates or clearly duplicates, it is very difficult to generate ambiguous cases that would help create a highly accurate classifier. Based on this observation, some duplicate detection systems used active learning methods to automatically locate such ambiguous pairs. Their method suggested that, by creating multiple classifiers, trained using slightly different data or parameters, it is possible to detect ambiguous cases and then ask the user for feedback. The key innovation in this work is the creation of several redundant functions and the concurrent exploitation of their conflicting actions in order to discover new kinds of inconsistencies among duplicates in the data set.

2.5 Distance based Techniques

Even active learning techniques require some training data or some human effort to create the matching models. In the absence of such training data or the ability to get human input, supervised and active learning techniques are not appropriate. One way of avoiding the need for training data is to define a distance metric for records which does not need tuning through training data. Distance-based approaches[6], that conflate each record in one big field may ignore important information that can be used for duplicate detection. A simple approach is to measure the distance between individual fields, using the appropriate distance metric for each field, and then compute the weighted distance between the records. In this case, the problem is the computation of the weights and the overall setting becomes very similar to the probabilistic setting. one of the problems of the distance-based techniques is the need to define the appropriate value for the matching threshold. In the presence of training data, it is possible to find the appropriate threshold value. However, this would nullify the major advantage of distance-based techniques, which is the ability to operate without training data. Recently, Chaudhuri et al. [9] proposed a new framework for distance-based duplicate detection, observing that the distance thresholds for detecting real duplicate entries are different from each database tuple. To detect the appropriate threshold, Chaudhuri et al. observed that entries that correspond to the same real-world object but have different representation in the database tend 1) to have small distances from each other (compact set property), to have only a small number of other neighbors within a small distance (sparse neighborhood property). Furthermore, Chaudhuri et al. propose an efficient algorithm for computing the required threshold for each object in the database and show that the quality of the results outperforms approaches that rely on a single, global threshold.

2.6 Rule Based Approaches

A special case of distance-based approaches [6] and [7] is the use of rules to define whether two records are the same or not. Rule-based approaches can be considered as distance-based techniques, where the distance of two records is either 0 or 1. It is noteworthy that such rule-based approaches which require a human expert to devise meticulously crafted matching rules typically result in systems with high accuracy. However, the required tuning requires extremely high manual effort from the human experts and this effort makes the deployment of such systems difficult in practice. Currently, the typical approach is to use a system that generates matching rules from training data and then manually tune the automatically generated rules. The mapping transformation standardizes data, the matching transformation finds pairs of records that probably refer to the same real object, the clustering transformation groups together matching pairs with a high similarity value, and, finally, the merging transformation collapses each individual cluster into a tuple of the resulting data source. It is noteworthy that such rule-based approaches which require a human expert to devise meticulously crafted matching rules typically result in systems with high accuracy. However, the required tuning requires extremely high manual effort from the human experts and this effort makes the deployment of such systems difficult in practice. Currently, the typical approach is to use a system that generates matching rules from training data and then manually tune the automatically generated rules.

2.7 Unsupervised learning

One way to avoid manual labeling of the comparison vectors is to use clustering algorithms and group together similar comparison

vectors. The idea behind most unsupervised learning approaches[4] and [7] for duplicate detection is that similar comparison vectors correspond to the same class. The idea of unsupervised learning for duplicate detection has its roots in the probabilistic model proposed by Fellegi and Sunter [8]. The idea of unsupervised learning for duplicate detection has its roots in the probabilistic model proposed by Fellegi and Sunter . When there is no training data to compute the probability estimates, it is possible to use variations of the Expectation Maximization algorithm to identify appropriate clusters in the data. Verykios et al. [10] propose the use of a bootstrapping technique based on clustering to learn matching models. The basic idea, also known as cotraining [11], is to use very few labeled data, and then use unsupervised learning techniques to appropriately label the data with unknown labels. Initially, Verykios et al. treat each entry of the comparison vector (which corresponds to the result of a field comparison) as a continuous, real variable. The basic premise is that each cluster contains comparison vectors with similar characteristics. Therefore, all the record pairs in the cluster belong to the same class (matches, nonmatches, or possiblematches).

There are multiple techniques for duplicate record detection. We can divide the techniques into two broad categories: ad hoc techniques that work quickly on existing relational databases and more "principled" techniques that are based on probabilistic inference models. While probabilistic methods outperform ad hoc techniques in terms of accuracy, the ad hoc techniques work much faster and can scale to databases with hundreds of thousands of records. Probabilistic inference techniques are practical today only for data sets that are one or two orders of magnitude smaller than the data sets handled by ad hoc techniques. A promising direction for future research is to devise techniques that can substantially improve the efficiency of approaches that rely on machine learning and probabilistic inference. A question that is unlikely to be resolved soon is the question of which of the presented methods should be used for a given duplicate detection task. Unfortunately, there is no clear answer to this question. The duplicate record detection task is highly data-dependent and it is unclear if we will ever see a technique dominating all others across all data sets. The problem of choosing the best method for duplicate data detection is very similar to the problem of model selection and performance prediction for data mining.

3. IMPROVING THE EFFICIENCY OF DUPLICATE DETECTION

So far, in our discussion of methods for detecting whether two records refer to the same real-world object, we have focused mainly on the quality of the comparison techniques and not on the efficiency of the duplicate detection process. Now, we turn to the central issue of improving the speed of duplicate detection. An elementary technique for discovering matching entries in tables A and B is to execute a "nested-loop" comparison, i.e., to compare every record of table A with every record in table B. Unfortunately, such a strategy requires a total of |A| . |B| comparisons, a cost that is prohibitively expensive even for moderately sized tables. We describe techniques that substantially reduce the number of required comparisons. Another factor that can lead to increased computation expense is the cost required for a single comparison. It is not uncommon for a record to contain tens of fields. Therefore, each record comparison requires multiple field comparisons and each field comparison can be expensive.

3.1 Blocking

One "traditional" method for identifying identical records in a database table is to scan the table and compute the value of a hash function for each record. The value of the hash function defines the "bucket" to which this record is assigned. By definition, two records that are identical will be assigned to the same bucket. Therefore, in order to locate duplicates, it is enough to compare only the records that fall into the same bucket for matches. The hashing technique cannot be used directly for approximate duplicates since there is no guarantee that the hash value of two similar records will be the same. However, there is an interesting counterpart of this method, named blocking. As discussed above with relation to utilizing the hash function, blocking typically refers to the procedure of subdividing files into a set of mutually exclusive subsets (blocks) under the assumption that no matches occur across different blocks. A common approach to achieving these blocks is to use a function such as Soundex, NYSIIS, or Metaphone

3.2 Sorted Neighborhood Approach

The method consists of the following three steps:

- Create key: A key for each record in the list is computed by extracting relevant fields or portions of fields.
- Sort data: The records in the database are sorted by using the key found in the first step. A sorting key is defined to be a sequence of attributes, or a sequence of substrings within the attributes, chosen from the record in an ad hoc manner. Attributes that appear first in the key have a higher priority than those that appear subsequently.
- Merge: A fixed size window is moved through the sequential list of records in order to limit the comparisons for matching records to those records in the window.

If the size of the window is w records, then every new record that enters that window is compared with the previous w -1 records to find "matching" records. The first record in the window slides out of it. The sorted neighborhood approach relies on the assumption that duplicate records will be close in the sorted list, and therefore will be compared during the merge step. The effectiveness of the sorted neighborhood approach is highly dependent upon the comparison key that is selected to sort the records. In general, no single key will be sufficient to sort the records in such a way that all the matching records can be detected. If the error in a record occurs in the particular field or portion of the field that is the most important part of the sorting key, there is a very small possibility that the record will end up close to a matching record after sorting.

3.3 Clustering and Canopies

To improve the performance of a basic "nested-loop" record comparison by assuming that duplicate detection is transitive. Under the assumption of transitivity, the problem of matching records in a database can be described in terms of determining the connected components of an undirected graph. At any time, the connected components of the graph correspond to the transitive closure of the "record matches" relationships discovered so far.

We use a union-find structure to efficiently compute the connected components of the graph. During the Union step, duplicate records are "merged" into a cluster and only a "representative" of the cluster is kept for subsequent comparisons. This reduces the total number of record comparisons without substantially reducing the accuracy of the duplicate detection process. The concept behind this approach is that, if a record b is not similar to a record b already in the cluster, then it will not match the other members of the cluster either. The use of canopies for speeding up the duplicate detection process. The basic idea is to use a cheap comparison metric to group records into overlapping clusters called canopies. (This is in contrast to blocking that requires hard, nonoverlapping partitions.) After the first step, the records are then compared pairwise, using a more expensive similarity metric that leads to better qualitative results. The assumption behind this method is that there is an inexpensive similarity function that can be used as a "quick-and-dirty" approximation for another, more expensive function. For example, if two strings have a length difference larger than 3, then their edit distance cannot be smaller than 3. In that case, the length comparison serves as a cheap (canopy) function for the more expensive edit distance.

4. PROPOSED APPROACH

The focus is on the Web databases from the same domain i.e., Web databases that provide the same type of records in response to user queries. Different users may have different criteria for what constitute a duplicate even for records within the same domain. 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 examples, to our knowledge all works assume that the positive examples are correct. Two classifiers along with Gaussian mixture model identifies the duplicate vector set iteratively.

4.1 Weighted Component Similarity Summing Classifier

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 Naive Bayes, 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. 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 The intuition for the weight assignment includes, The similarity between two duplicate records should be close to 1. For a duplicate vector V12 that is formed by a pair of duplicate records r1 and r2, we need to assign large weights to the components with large similarity values and small weights to the components with small similarity values . The similarity for two nonduplicate records should be close to 0. Hence, for a nonduplicate vector V12 that is formed by a pair of nonduplicate records r1 and r2, we need to assign small weights to the components with large similarity values and large weights to the components with small similarity values.

4.2 Support Vector Machine Classifier

After detecting a few duplicate vectors whose similarity scores are bigger than the threshold using the WCSS classifier[12], 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 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 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.

Similarity-based classifiers estimate the class label of a test sample based on the similarities between the test sample and a set of labeled training samples, and the pairwise similarities between the training samples. Like others, we use the term similarity-based classification whether the pairwise relationship is a similarity or dissimilarity. Similarity-based classification does not require direct access to the features of the samples, and thus the sample space can be any set, not necessarily a Euclidean space, as long as the similarity function is well defined for any pair of samples.

4.3 Gaussian Mixture Model

The complete Gaussian mixture model [15], is parameterized by the mean vectors μ_i, covariance matrices $\sum_i$ and mixture weights w_i from all component densities. These parameters are collectively represented by the notation,

$$\lambda = \{w_i, \mu_i, \sum_i\} \quad i = 1, \ldots, M. \qquad (2)$$

There are several variants on the GMM shown in Equation (2). The covariance matrices, $\sum_i$, can be full rank or constrained to be diagonal. Additionally, parameters can be shared, or tied, among the Gaussian components, such as having a common covariance matrix for all components, The choice of model configuration (number of components, full or diagonal covariance matrices, and parameter tying) is often determined by the amount of data available for estimating the GMM parameters. The use of a GMM for representing feature distributions in a biometric system may also be motivated by the intuitive notion that the individual component densities may model some underlying set of hidden classes. For example, in speaker recognition, it is reasonable to assume the acoustic space of spectral related features corresponding to a speaker's broad phonetic events, such as vowels, nasals or fricatives.

4.4 Evaluation Metric

The performance of the duplicate detected dataset [13],[14] can be reported using precision and recall, which are defined as follows:

$$\text{Precision} = \frac{\#\text{of correctly Identified Duplicate Pairs}}{\#\text{of All Identified Duplicate Pairs}}$$

$$\text{Recall} = \frac{\#\text{of correctly Identified Duplicate Pairs}}{\#\text{of True duplicate Pairs}}$$

Due to the usually imbalanced distribution of matches and nonmatches in the weight vector set, these commonly used accuracy measures are not suitable for assessing the quality of record matching. The large number of nonmatches usually dominates the accuracy measure and yields results that are too optimistic. To measure the performance F-measure can also be used, which is the harmonic mean of precision and recall, to evaluate the classification quality:

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

5. EXPERIMENTS

We evaluate our model on dataset of research paper citations. The dataset is from the Cora Computer Science Research Paper Engine, containing about 1800 citations, with 600 unique papers and 200 unique venues. The datasets are manually labeled for both paper coreference and venue coreference, as well as manually segmented into fields, such as author, title, etc. The data were collected by searching for certain authors and topic, and they are split into subsets with non-overlapping papers for the sake of cross-validation experiments. We used a number of feature functions, including exact and approximate string match on normalized and unnormalized values for the following citation fields: title, booktitle, journal, authors, venue, date, editors, institution, and the entire unsegmented citation string. We also calculated an unweighted cosine similarity between tokens in the title and author fields. Additional features include whether or not the papers have the same publication type (e.g. journal or conference), as well as the numerical distance between fields such as year and volume. All real values were binned and converted into binary-valued features. To evaluate performance, we compare the clusters output by our system with the true clustering using pairwise metrics and use the metrics ratios for evaluation. The Cora web site(www.cora.whizbang.com)provides a search interface to over 50,000 computer science research papers . As part of the site's functionality, we provide an interface for traversing the citation graph. That is, for a given paper, we provide links to all other papers it references, and links to all other papers that reference it, in the respective bibliography section of each paper. To provide this interface to the data, it is necessary to recognize when two citations from different papers are referencing the same third paper, even though the text of the citations may differ. For example, one paper may abbreviate firrst author names, while the second may include them.

The datasets were collected and the algorithm was coded in Java and run on a machine with Windows XP operating system. The duplicates are identified and the core Gaussian Mixture Model is being implemented.

6. REFERENCES

[1] W.Su, J. Wang, and Frederick H. Lochovsky, "Record Matching over Query Results from Multiple Web Databases" IEEE Transactions on knowledge and data engineering.

[2] R. Baxter, P. Christen, and T. Churches, "A Comparison of Fast Blocking Methods for Record Linkage," Proc. KDD Workshop Data Cleaning, Record Linkage, and Object Consolidation, pp. 25-27, 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] O. Bennjelloun, H. Garcia-Molina, D. Menestrina, Q. Su,S.E.Whang, and J. Widom, "Swoosh: A Generic Approach to Entity Resolution," The VLDB J., vol. 18, no. 1, pp. 255-276, 2009.

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

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

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

[9] S. Chaudhuri, V. Ganti, and R. Motwani, "Robust Identification of Fuzzy Duplicates," Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05),pp. 865-876, 2005.

[10] V.S. Verykios, A.K. Elmagarmid, and E.N. Houstis, "Automating the Approximate Record Matching Process," Information Sciences, vol. 126, nos. 1-4, pp. 83-98, July 2000.

[11] A. Blum and T. Mitchell, "Combining Labeled and Unlabeled Data with Co-Training," COLT '98: Proc. 11th Ann. Conf. Computational Learning Theory, pp. 92-100, 1998.

[12] 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.

[13] P. Christen and K. Goiser, "Quality and Complexity Measures for Data Linkage and Deduplication," Quality Measures in Data Mining, F. Guillet and H. Hamilton, eds., vol. 43, pp. 127-151, Springer, 2007.

[14] W.W. Cohen, H. Kautz, and D. McAllester, "Hardening Soft Information Sources," Proc. ACM SIGKDD, pp. 255-259, 2000.

[15] Gaussian Mixture Models Douglas Reynolds MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 02140, USA

[16] 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.