Effective Duplicate Detection Using Generational Evolutionary Algorithm

M.Chittibabu*1, B.V.Ramana*2, S.Kavitha*3
M.Tech (CSE) Student, Department of CSE, HIT, Keesara, D.t: Ranga Reddy, AP, India
Assistant Professor, Department of CSE, HIT, Keesara, D.t: Ranga Reddy, AP, India
Professor, Department of CSE, HIT, Keesara, D.t: Ranga Reddy, AP, India

ABSTRACT:
The record deduplication is the task of identifying, in a data repository, records that refer to the same real world entity or object in spite of misspelling words, types, different writing styles or even different schema representations or data types. In existing system aims at providing Unsupervised Duplication Detection (UDD) method which can be used to identify and remove the duplicate records from different data sources. Starting from the non duplicate set, the two cooperating classifiers, a Weighted Component Similarity Summing Classifier (WCSS) and Support Vector Machine (SVM) are used to iteratively identify the duplicate records from the non duplicate record and present a genetic programming (GP) approach to record deduplication. Their GP-based approach is also able to automatically find effective deduplication functions. We propose to employ learnable 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 learnable 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 datasets show that our framework can improve duplicate detection accuracy over traditional techniques.

KEYWORDS: SVM, UDD, GP based approach, de-duplication function.

I. INTRODUCTION
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. Variations are particularly pronounced in data that is automatically extracted from unstructured or semi-structured documents or web pages. 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. Some previous work has addressed the problem of identifying duplicate records, where it was referred to as record linkage, the merge/purge problem, duplicate detection, hardening soft databases, reference matching and entity name clustering and match. Typically, standard string similarity metrics such as edit distance or vector-space cosine similarity are used to determine whether two values or records are alike enough to be duplicates. Some more recent work has investigated the use of pairing functions that combine multiple standard metrics. Because an estimate of similarity between strings can vary significantly depending on the domain and specific field under consideration, traditional similarity measures may fail to estimate string similarity correctly. At the token level, certain words can be informative when comparing two strings for equivalence, while others are ignorable. For example, ignoring the substring “Street” may be acceptable when comparing addresses, but not when comparing names.
of people (e.g. “Nick Street”) or newspapers (e.g. “Wall Street Journal”). At the character level, certain characters can be consistently replaced by others or omitted when syntactic variations are due to systematic typographical or OCR errors. Thus, accurate similarity computations require adapting string similarity metrics for each field of the database with respect to the particular data domain. Rather than hand-tuning a distance metric for each field, we propose to use trainable similarity measures that can be learned from small corpora of labeled examples, and thus adapt to different domains.

We present two such string similarity measures. The first one utilizes the Expectation-Maximization (EM) algorithm for estimating the parameters of a generative model based on string edit distance with affine gaps. The other string similarity measure employs a Support Vector Machine (SVM) [24] to obtain a similarity estimate based on the vector-space model of text. The character based distance is best suited for shorter strings with minor variations, while the measure based on vector-space representation is more appropriate for fields that contain longer strings with more global variations. Our overall system, MARLIN (Multiply Adaptive Record Linkage with INduction), employs a two-level learning approach. First, string similarity measures are trained for every database field so that they can provide accurate estimates of string distance between values for that field. Next, a final predicate for detecting duplicate records is learned from similarity metrics applied to each of the individual fields. We again utilize Support Vector Machines for this task, and show that they outperform decision trees, the classifier used in prior work [23, 22]. We evaluate our approach on several real-world data sets containing duplicate records and show that MARLIN can lead to improved duplicate detection accuracy over traditional techniques.

II. RELATED WORK

There arise so many problems when data collected from different sources are to be used since these data uses different styles and standards. Moreover replica of documents is made for Optical Character Recognition (OCR) documents. This lead to inconsistencies among the data stored in repositories. The problem becomes more complicated when a user needs to obtain user-specified information from huge volume of data stored in large databases like repositories. To solve these issues, information from unstructured data is to be extracted and stored in databases with perfect structure. This enables user to obtain information retrieval with increased speed and accuracy. The common problems met are:

1) The existing structured databases of entities are organized very differently from labeled unstructured text.
2) There is significant format variation in the names of entities in the database and the unstructured text. 3) In most cases the database will be large whereas labeled text data will be small. Features designed from the databases should be efficient to apply and should not dominate features that capture contextual words and positional information from the limited labeled data.

To address these issues, the data integration system is designed. This system uses Semi-Markov models for extracting information from structured data and labeled unstructured data in spite of their format, structure and size variations.

The former method is enhanced by a semi automatic extraction method using DEG. It follows the following three steps.

1) To gather the necessary knowledge and then transform them into useable form. The knowledge can be obtained from any source such as encyclopedia, a traditional relational
database. A general ontology like Mikrokosmos, etc. This
needs to handle data in different formats.

2) Automatically generate an initial data-extraction ontology
based on the acquired knowledge and sample target
documents. Gathered knowledge is transformed into
Extensible Markup Language (XML) format and various
XML documents are combined to produce a high level
schema. This schema defines the set of attributes that may
appear in generated data extraction ontology.

3) Finally user validates the initial data extraction ontology
which is generated using set of validation documents. If the
result is not satisfactory, user applies Ontology Editor to the
generated ontology. The Ontology Editor provides a method
of editing an Object Relationship Model (ORM) and its
associated data frames and also provides debugging
functionality for editing regular expressions in data frames
by displaying sample text with highlighting on sample
source documents.

Even though Database Enhancement Gateway (DEG)
method is efficient, it requires human validation. Thus a
fully automatic method is proposed which uses tag path
clustering. Usually the list of objects is extracted from
databases using pair wise similarity match. But this pair wise
similarity match did not address the nested data structures or
more complicated structure. Hence the tag path clustering
focuses on how a distinct tag path (i.e., a path from the root
to a leaf in the DOM tree) appears repeatedly in
the document. The occurrence of a pair of tag path patterns
(called visual signals) is compared to estimate how likely
these two tag paths represent the same list of objects.
Comparison is done using a similarity measure which uses a
similarity function which captures how likely two visual
signals belong to the same data region.

III. Genetic Programming

Genetic Programming (GP), an inductive learning technique
introduced by Koza in as an extension to Genetic
Algorithms (GA), is a problem-solving system inspired by
the idea of Natural Selection. The search space of a problem,
i.e., the space of all possible solutions to the problem, is
investigated using a set of optimization techniques that
imitate the theory of evolution, combining natural selection
and genetic operations to provide a way to search for the
fittest solution. The evolution process starts with an initial
population composed by a set of individuals. Generally, the
initial population is generated randomly. Each individual
denotes a solution to the examined problem and is
represented by a tree. To each individual is associated a
fitness value. This value is determined by an evaluation
function, also known as fitness function. The fitness value
indicates goodness of an individual and it is used to
eliminate from the populations all “unfit” individuals,
selecting only those that are closest to the desired goal. The
individuals will evolve generation by generation through
genetic operations such as reproduction, crossover, and
mutation. The reproduction operator simply breeds a new
individual. The mutation operator simulates the deviations
that take place in the reproduction process. Finally, the
crossover operator generates new individuals by the
composite of some characteristics present in two other
individuals (the parents).

Thus, for each generation, after the genetic operations are
applied, a new population replaces the current one. The
fitness value is measured for each new individual, and the
process is repeated over many generations until the
termination criterion has been satisfied. This criterion can be
a reestablished maximum number of generations or some
IV. MODELLING DEDUPLICATION AND ITS ANALYSIS

Data Cleaning is a time consuming process because of its lengthy activities. Since data preparation is done from multiple sources, there precedes data redundancies which brings problem in data storage capacity, processing capacity and also manual vagueness to maintainability. Deduplication is a specialized data compression technique for eliminating coarse-grained redundant data. The technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent across a link. There are multiple techniques for improving the efficiency and scalability of approximate duplicate detection algorithms.

4.1. Active-Learning Techniques

The main task that must be carried out is to project a function that must be able to resolve when a pair of records refers to the same entity in spite of various data inconsistencies. The earlier function to resolve was hand coded function where requires manually searching for various data inconsistencies between any two records spread apart in large lists, which the task very non-trivial and challenging. Learning-based deduplication system was introduced which discovered challenging training pairs using method called Active learning [8]. The designed technique is a learning based deduplication system that allows automatic construction of the deduplication function by using a novel method of interactively discovering challenging training pairs. In this method the learner is automated to do the difficult task of of bringing together the potentially confusing record pairs. So the user has to only perform the easy task of labeling the selected pairs of records as duplicate or not. The system for deduplication consist of three primary inputs they are:

a) Database of records (D) The original set D of records in which duplicates need to be detected.

b) Initial training pairs (L) An optional small (less than ten) seed L of training records arranged in pairs of duplicates or non-duplicates.

c) Similarity functions (F) A set F of functions each of which computes a similarity match between two records based on any subset of d attributes.

![Fig. 1 Overall design and working of Active-learning-based technique [8]](image)

The main idea behind this system is that most duplicate and non-duplicate pairs are clearly distinct. The system starts with small subsets of pairs of records designed for training which have been characterized as either matched or unique. This initial set of labeled data forms the training data for a preliminary classifier. In the sequel, the initial classifier is used for predicting the status of unlabeled pairs of records. The initial classifier will make clear determinations on some unlabeled instances but lack determination on most. The goal is to seek out from the unlabeled data pool those instances which, when labeled, will improve the accuracy of the classifier at the fastest possible rate. Pairs whose status is difficult to determine serve to strengthen the integrity of the learner. Conversely, instances in which the learner can easily predict the status of the pairs do not have much effect on the
learner. Using this technique, Active-learning-based system can quickly learn the peculiarities of a data set and rapidly detect duplicates using only a small number of training data [1]. Active-learning-based system is not appropriate in some places because it always requires some training data or some human effort to create the matching models.

V. RECORDLEVEL SIMILARITY

5.1 Combining similarity across multiple fields
When the distance between records composed of multiple fields is calculated, it is necessary to combine similarity estimates from individual fields in a meaningful manner. Because correspondence between overall record similarity and single-field similarity can vary greatly depending on how informative the fields are, it is necessary to weight fields according to their contribution to the true distance between records.

While statistical aspects of combining similarity scores for individual fields have been addressed in previous work on record linkage [25], availability of labeled duplicates allows a more direct approach that uses a binary classifier that computes a “pairing function” [4]. Given a database that contains records composed of $k$ different fields and a set $D = \{d_1(\cdot, \cdot), \ldots, d_m(\cdot, \cdot)\}$ of distance metrics, we can represent any pair of records by an $mk$-dimensional vector. Each component of the vector represents similarity between two field values of the records that is calculated using one of the $m$ distance metrics. Matched pairs of duplicate records can be used to construct a training set of such feature vectors by assigning them a positive class label. Pairs of records that are not labeled as duplicates implicitly form the complementary set of negative examples. If the transitive closure of matched pairs contains disjoint sets of duplicate records, this approach will result in noisy negative examples. Next, a binary classifier is trained using these training vectors to discriminate between pairs of records corresponding to duplicates and non-duplicates. Overall, this approach follows the same framework that is used for learnable vector-space string similarity in the previous section. Following the same reasoning, SVMs are a good classifier choice due to their resilience to noise and ability to handle correlated features well. The distance from the hyperplane provides a measure of confidence in the pair of records being a duplicate; it can be transformed to an actual similarity value using Eq.(4).

Fig.5 illustrates this process of computing record similarity using multiple similarity measures over each field and a binary classifier to categorize the resulting feature vector as belonging to the class of duplicates or non-duplicates, resulting in a distance estimate. For each field of the database, two learnable distance measures, $d_1$ and $d_2$, are trained and used to compute similarity for that field. The values computed by these measures form the feature vector that is then classified by a support vector machine, producing a confidence value that represents similarity between the database records.

5.2 The overall duplicate detection framework
An overall view of our system, MARLIN, is presented in Fig.6. The training phase consists of two steps. First, the learnable distance metrics are trained for each record field. The training corpus of paired field-level duplicates and non-duplicates is obtained by taking pairs of values for each field from the set of paired duplicate records. Because duplicate records may contain individual fields that are not equivalent, training data can be noisy. For example, if one record describing a restaurant contains ‘Asian’ in the Cuisine field, and a duplicate record contains ‘Seafood’, a noisy training pair is formed that implies equivalence between these two descriptors. However, this issue does not pose a serious
problem for our approach for two reasons. First, particularly noisy fields that are unhelpful for identifying record-level duplicates will be considered irrelevant by the classifier that combines similarities from different fields. Second, the presence of such pairs in the database indicates that there is a degree of similarity between such values, and using them in training allows the learnable metric to capture that likelihood.

After individual similarity metrics are learned, they are used to compute distances for each field of duplicate and non-duplicate record pairs to obtain training data for the binary classifier in the form of vectors composed of distance features. The duplicate detection phase starts with generation of potential duplicate pairs. Given a large database, producing all possible pairs of records and computing similarity between them is too expensive since it would require $O(n^2)$ distance computations. MARLIN utilizes the canopies clustering method [13] using Jaccard similarity, a computationally inexpensive metric based on an inverted index, to separate records into overlapping clusters (“canopies”) of potential duplicates. Pairs of records that fall in each cluster then become candidates for a full similarity comparison shown in Fig.5. Learned distance metrics are then used to calculate distances for each field of each pair of potential duplicate records, thus creating distance feature vectors for the classifier. Confidence estimates for belonging to the class of duplicates are produced by the binary classifier for each candidate pair, and pairs are sorted by increasing confidence.

VI. CONCLUSION

Duplicate detection is an important problem in data cleaning, and an adaptive approach that learns to identify duplicate records for a specific domain has clear advantages over static methods. Experimental results demonstrate that trainable similarity measures are capable of learning the specific notion of similarity that is appropriate for a specific domain. We presented two learnable distance measures that improve over character-based and vector-space based metrics and allow specializing them for specific datasets using labeled examples. We have also shown that support vector machines can be effectively utilized for some datasets both for string similarity and record similarity computations, outperforming traditional methods; we hope to improve on these initial results in our future work. Our overall framework for duplicate detection integrates previous work on adaptive methods with learnable similarity measures, leading to improved results.

VII. REFERENCES


