IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 10, Issue 2 (Mar. - Apr. 2013), PP 22-27
www.iosrjournals.org

Efficient Record De-Duplication Identifying Using Febrl Framework

K.Mala[1], Mr. S.Chinnadurai[2]
[1]*Student, M.E. (C.S.E), Srinivasan Engineering College, Perambalur-621212*
[2]*Assistant Professor, Department Of Cse, Srinivasan Engineering College, Perambalur -621212*

Abstract *: Record linkage is the problem of identifying similar records across different data sources. The similarity between two records is defined based on domain-specific similarity functions over several attributes. De-duplicating one data set or linking several data sets are increasingly important tasks in the data preparation steps of many data mining projects. The aim is to match all records relating to the same entity. Different measures have been used to characterize the quality and complexity of data linkage algorithms, and several new metrics have been proposed. An overview of the issues involved in measuring data linkage and de-duplication quality and complexity. A matching tree is used to overcome communication overhead and give matching decision as obtained using the conventional linkage technique. Developed new indexing techniques for scalable record linkage and de-duplication techniques into the febrl framework, as well as the investigation of learning techniques for efficient and accurate indexing.*

Keywords *- data cleaning; similarity matching; record linkage; data mining pre-processing; febrl.*

I. INTRODUCTION

The most recent have observe a marvelous increase in the use of computerized databases for supporting a variety of company decisions. The data needed to support these decisions are often spread in diverse dispersed databases. In such cases, it may be necessary to linkage records in multiple databases so that one can merge and use the data pertaining to the same real world entity. If the databases use the same set of design standards, this linking can easily be done using the primary Key, however, since these heterogeneous databases are usually designed and managed by different organizations, there may be no common candidate key for linking the records. Although it may be possible to use common non key attributes (such as name, address, and date of birth) for this purpose, the result obtained using these attributes may not always be accurate.

The databases exhibiting entity heterogeneity are distributed, and it is not possible to create and maintain a central data repository or warehouse where pre-computed linkage results can be stored. A centralized solution may be impractical for several reasons. First, if the databases span several organizations, the ownership and cost allocation issues associated with the warehouse could be quite difficult to address. Second, even if the warehouse could be developed, it would be difficult to keep it up-to-date. As updates occur at the operational databases, the linkage results would become stale if they are not updated immediately. This staleness maybe unacceptable in many situations. For instance, in a criminal investigation, one maybe interested in the profile of crimes committed in the last 24 hours within a certain radius of the crime scene. In order to keep the warehouse current, the sites must agree to transmit incremental changes to the data warehouse on a real-time basis. Even if such an agreement is reached, it would be difficult to monitor and enforce it. For example, a site would often have no incentive to report the insertion of a new record immediately. Therefore, these changes are likely to be reported to the warehouse at a later time, thereby increasing the staleness of the linkage tables and limiting their usefulness. In addition, the overall data management tasks could be prohibitively time-consuming, especially in situations where there are many databases, each with many records, undergoing real-time changes.

II. RELATED WORK

DATA CLEANING AND RECORD LINKAGE PROCESS

A general schematic outline of the record linkage process is given As most real-world data collections contain noisy, incomplete and incorrectly formatted information, data cleaning and standardization are important pre-processing steps for successful record linkage, and also before data can be loaded into data warehouses or used for further analysis or data mining. A lack of good quality data can be one of the biggest obstacles to successful record linkage and de-duplication. The main task of data cleaning and standardization is the conversion of the raw input data into well defined, consistent forms, as well as the resolution of inconsistencies in the way information is represented and encoded.

If two databases, A and B, are to be linked, potentially each record from A has to be compared with all records from B. The total number of potential record pair comparisons thus equals the product of the size of the two databases, $|A| \times |B|$, with $|\cdot|$ denoting the number of records in a database. Similarly, when de-duplicating a

database, A, the total number of potential record pair comparisons is $|A| \times (|A| - 1)/2$, as each record potentially has to be compared with all others. The performance bottleneck in a record linkage or de-duplication system is usually the expensive detailed comparison of record fields (or attributes) between pairs of records, making it unfeasible to compare all pairs when the databases are large. Assuming there are no duplicate records in the databases (i.e. one record in database A can only match to one record in database B, and vice versa), then the maximum number of true matches corresponds to the number of records in the smaller database. Therefore, while the computational efforts increase quadratic ally, the number of potential true matches only increases linearly when linking larger databases. This also holds for de-duplication, where the number of duplicate records is always less than the total number of records in a database.

To reduce the large amount of potential record pair comparisons, record linkage methods employ some form of indexing or filtering techniques, collectively known as blocking. a single record field (attribute) or a combination of fields, often called the blocking key, is used to split the databases into blocks. All records that have the same value in the blocking key will be inserted into one block, and candidate record pairs are then generated only from records within the same block. These candidate pairs are compared using a variety of comparison functions applied to one or more (or a combination of) record fields. These functions can be as simple as an exact string or a numerical comparison, can take variations and typographical errors into account, or can be as complex as a distance comparison based on look-up tables of geographic locations (longitudes and latitudes). Each field comparison returns a numerical similarity value, called a matching weight, often in normalized form. Two field values that are equal, therefore, will have a matching weight of 1, while the matching weight of two completely different field values will be 0. Field values that are somewhat similar will have a matching weight somewhere between 0 and 1. A weight vector is formed for each compared record pair containing all the matching weights calculated by the different comparison functions. These weight vectors are then used to classify record pairs into matches, non-matches, and possible matches, depending upon the decision model. Record pairs that were removed by the blocking process are classified as non-matches without being compared explicitly. A variety of evaluation measures can then be used to assess the quality of the linked record pairs.

III. PROBLEM DEFINITION

FEBRL FRAMEWORK

Python is an ideal platform for rapid prototype development as it provides data structures such as sets, lists and dictionaries (associative arrays) that allow efficient handling of very large data sets, and includes many modules offering a large variety of functionalities. For example, it has excellent built-in string handling capabilities, and the large number of extension modules facilitate, for example, database access and graphical user interface (GUI) development. For the Febrl user interface, the PyGTK4 library and the Glade5 toolkit were use, which, combined, allow rapid platform independent GUI development

Febrl is suitable for the rapid development, implementation, and testing of new and improved record linkage algorithms and techniques, as well as for both new and experienced users to learn about and experiment with various record linkage techniques.

3.1) Input Data Initialization

In a first step, a user has to select if she or he wishes to conduct a project for (a) cleaning and standardization of a data set, (b) de-duplication of a data set, or (c) linkage of two data sets. The 'Data' page of the Febrl GUI will change accordingly and either show one or two data set selection areas. Several texts based data set types are currently supported, including the most commonly used comma separated values (CSV) file format. SQL database access will be added in the near future. Various settings can be selected, such as if a data set file contains a header line with field names (if not these field names can be entered manually); if one of the fields contains unique record identifiers; a list of missing values can be given (like 'missing' or 'n/a') that automatically will be removed when the data is loaded; and there are data type specific parameters to be set as well (such as the delimiter for CSV data sets).

3.2) Data Exploration

The 'Explore' page allows the user to analyses the selected input data set(s) in order to get a better understanding of the content and quality of the data to be used for a standardization, de-duplication or linkage project. In order to speed up exploration of large data sets, it is possible to select a sampling rate as percentage of the number of records in a data set.

3.3) Data Cleaning and Standardisation

The cleaning and standardization of a data set using the Febrl GUI is currently done separately from a linkage or de-duplication project, rather than as a first step. A data set can be cleaned and standardized and is

written into a new data set, which in turn can then be de-duplicated or used for a linkage. When a user selects the 'Standardization' project type, and has initialized a data set on the 'Data' page, she or he can define one or more component standardizes on the 'Standardize'page.Currently, component standardizes are available in Febrl for names, addresses, dates, and telephone numbers. The name standardize uses a rule-based approach for simple names (such as those made of one given- and one surname only) in combination with a probabilistic hidden Markov model (HMM) approach for more complex names (Churches et al. 2002), while address standardization is fully based on a HMM approach (Christen and Belacic 2005). These HMMs currently have to be trained outside of the Febrl GUI, using separate Febrl modules. Dates are standardized using a list of format strings that provide the expected formats of the dates likely to be found in the un cleaned input data set. Telephone numbers are also standardized using a rules based approach. Each standardize requires one or several input fields from the input data set (shown on the left side of a standardize in the GUI), and cleans and segments a component into a number of output fields (three for dates, five for phone numbers, six for names, and 27 for addresses), shown on the right side in the GUI.

IV. The Proposed Method

4.1 INDEXING DEFINITION

'QGramIndex', which uses sub-strings of length q (for example bigrams, where q = 2) to allow fuzzy blocking (Baxter et al. 2003); 'Canopy Index', which employs overlapping canopy clustering using TF-IDF or Jaccard similarity (Cohen and Richman 2002); 'String Map Index', which maps the index key values into a multi-dimensional space and performs canopy clustering on these multi-dimensional objects (Jin et al. 2003); and 'Suffix Array Index', which generates all suffixes of the index key values and inserts them into a sorted array to enable efficient access to the index key values and generation of the corresponding blocks (Aizawa and Oyama 2005).

For deduplication using 'BlockingIndex', 'Sorting Index' or 'QGram Index', the indexing step can be performed in an overlapping fashion with the field comparison step, by building an inverted index data structure while records are read from the input data set and their blocking key values are extracted and inserted into the index. The current record is compared with all previously read and indexed records having the same blocking key value. This approach can be selected by the user by ticking the 'De-duplication' indexing box. For a linkage, and using one of the three indexing methods mentioned above, the Big Match (Yancey 2002) approach can be selected, where first the smaller input data set is loaded and the inverted index data structures are built in main memory, including all record attribute values required in the comparison step. Each record of the larger input data set is then read, its blocking key values are extracted, and all records in the same block from the smaller data set are retrieved from the index data structure and compared with the current record. This approach performs only one single pass over the large data set and does not require indexing, sorting or storing of any of its records. The user can tick the corresponding 'Big Match' indexing box when conducting a linkage project.

4.2 FIELD COMPARISON FUNCTIONS

The comparison functions to be used to compare the field values of record pairs can be selected and setup on the 'Comparison'. Each field comparison requires the user to select one of the many available comparison functions as well as the two record fields that will be compared. While one normally would select fields with the same content from the two data sets (for example, to compare suburb names with suburb names), it is feasible to select different fields (for example to accommodate for swapped given- and surname values).

4.3 WEIGHT VECTOR CLASSIFICATION

The last major step required is the selection of the method used for weight vector classification and setting of its parameters. Currently, Febrl offers six different classification techniques. The simple 'Fellegi Sunter' classifier allows manual setting of two thresholds . With this classifier, the similarity weights of the weight vector of each compared record pair are summed into one matching weight, and record pairs that have a summed weight above the upper classification threshold are classified as matches, pairs with a matching weight below the lower threshold are classified as non-matches, and those record pairs that have a matching weight between the two classification thresholds are classified as possible matches.

With the 'Optimal Threshold' classifier it is assumed that the true match status for all compared record pairs is known (i.e. supervised classification), and thus an optimal threshold can be calculated based on the corresponding summed weight vectors.

The 'SuppVecMachine' classifier uses a supervised support vector machine (SVM) and thus requires the user to provide the true match status (as described above for the 'Optimal Threshold' classifier) of weight vectors in order to be able to train this classifier. It is based on the lib svm library, and the most important parameters of the SVM classifier can be set in the Febrl GUI. Finally, the 'Two Step' classifier is an unsupervised approach which in a first step selects weight vectors from the compared record pairs that with high

likelihood correspond to true matches and true non-matches, and in a second step uses these vectors as training examples for a binary classifier (Christen 2007). Several methods are implemented on how to select the training examples in the first step, and for the second step a SVM classifier or k-means clustering can be used. Experimental results have shown that this unsupervised approach to weight vector classification can a achieve linkage quality almost as good as fully supervised classification (Christen 2007).

V. Experimental Evaluation

The different methods were executed several times with different partition/window sizes. To obtain comparable results, the partition sizes for blocking were selected in such a way that there was always a corresponding window size with nearly the same total number of comparisons. This was achieved by sorting the tuples and cutting them in fixed size partitions. Additionally, an exhaustive comparison of all tuples was processed without any partitioning. This is especially interesting to see the impact of partitioning methods on the recall.

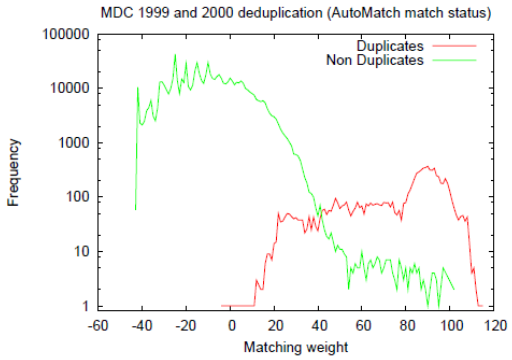

Fig 1: The density plot of the matching weights for a real-world administrative health data set. This plot is based on record pair comparison weights in a blocked comparison space. The smallest weight is -43, the highest 115. Note that the vertical axis with frequency counts is on a log scale

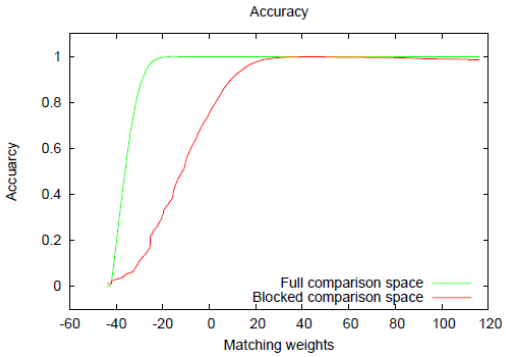

Fig 2a: Accuracy and matching weight

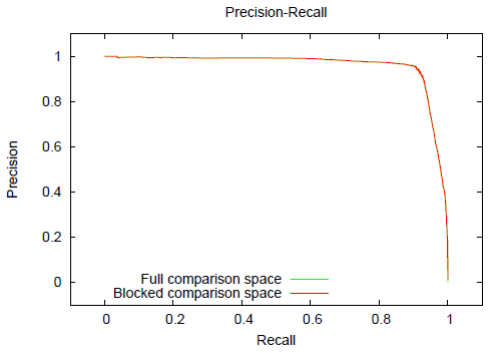

Fig 2b: precision and recall

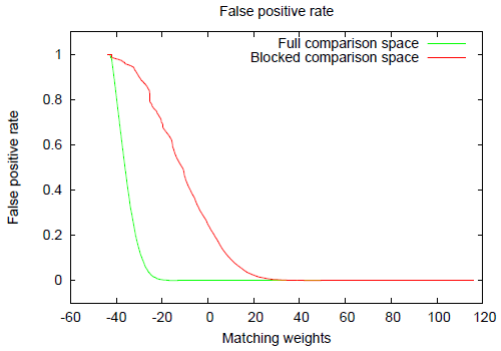

Fig 2c: matching weight and False Positive

Fig. 2. Quality measurements of a real world administrative health data set. The full comparison space (30; 698; 719; 310 record pairs) was simulated by assuming that the record pairs removed by blocking were normally distributed with matching weights between -43 and -10. Note that the precision-recall graph does not change at all, a ndthe F-measure graphs does change only slight. Accuracy and specificity are almost the same as both are dominated by the large number of true negatives. The ROC curve is the least illustrative graphs, which is again due to the large number of true negatives.

VI. Conclusion

Febrl is an training tool suitable for new record linkage users and practitioners, and to conduct small to medium sized experimental linkages and de-duplications with up to several hundred thousand records. Within the health sector, it can be used alongside commercial linkage systems for comparative linkage studies; and for both new and experienced record linkage practitioners to learn about the many advanced linkage techniques that have been developed in recent years and that are implemented in Febrl.

Acknowledgements

I take this chance to express my deep sense and knowledge of gratitude to our **Management**, our principal **Dr. B. Karthikeyan**, for providing an excellent infrastructure and support to pursue this research work at our college. I express my profound thanks to Head of the department **Prof. J. Mercy Geraldine** for his administration and keen interest, which motivated me along the course as well as research work, and also thank to all staff members.

The authors thank the anonymous referees for their extremely useful comments and suggestions on an earlier draft of this paper.

References

[1] Baxter.R, Christen.P, and Churches.T, "A Comparison of Fast Blocking Methods for Record Linkage," Proc. ACM Workshop Data Cleaning, Record Linkage and Object Consolidation (SIGKDD '03), pp. 25-27, 2003.

[2] Bilenko.M, Basu.S, and Sahami.M, "Adaptive Product Normalization: Using Online Learning for Record Linkage in Comparison Shopping," Proc. IEEE Int'l Conf. Data Mining (ICDM '05), pp. 58-65, 2005.

[3] Bilenko.M and Mooney.R.J, "On Evaluation and Training-Set Construction for Duplicate Detection," Proc. Workshop Data Cleaning, Record Linkage and Object Consolidation (SIGKDD '03), pp. 7-12, 2003.

[4] Bilenko.M, .Kamath.B, and Mooney.R.J, "Adaptive Blocking: Learning to Scale up Record Linkage," Proc. Sixth Int'l Conf. Data Mining (ICDM '06), pp. 87-96, 2006.

[5] Clark.D.E, "Practical Introduction to Record Linkage for Injury Research," Injury Prevention, vol. 10, pp. 186-191, 2004.

[6] Churches.T, Christen.P, .K Lim, and Zhu.J.X, "Preparation of Name and Address Data for Record Linkage Using Hidden Markov Models," Biomed Central Medical Informatics and Decision Making, vol. 2, no. 9, 2002. [7]Christen.P and Goiser.K, "Quality and Complexity Measures for Data Linkage and Deduplication," Quality Measures in Data Mining, ser. Studies in Computational Intelligence, Guillet.F and Hamilton.H, eds., vol. 43, Springer, pp. 127-151, 2007.

[8] Christen.P, "Febrl: An Open Source Data Cleaning, Deduplication and Record Linkage System With a Graphical User Interface," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '08), pp. 1065-1068, 2008.

[9] Christen.P, "Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '08), pp. 151-159, 2008.

[10] Peter christen,"A Survey of indexing Techniques for Scalable Record Linkage And Deduplication", IEEE Transcations on Knowledge And Data Engineering , vol 24,no.9.september 2012.

AUTHORS PROFILE

Mala.K received B.E degree in Computer Science Engineering from Vivekananda College, Tiruchengode, India in 2010. She is pursuing final year M.E in Computer Science Engg at Srinivasan Engg College. Her current research includes in Record Linkage and Deduplication and data structures. She is member of computer society India.

Chinnadurai.S received the B.Sc. and M.Sc. degree in Computer Science from Thanthai Hans Roever College, Perambalur. He has got the M.Tech. degree in Computer Science Engineering from Prist University, Thanjavur, India. He is an Asst.prof in Srinivasan Engg College, Perambalur. His research works includes wireless sensor networks and advance Ad-Hoc networks. He is a member of computer society India. He published journals about wireless sensor networks in the year 2012. He got the best content award in the conference sponsored by Institute of Research and Development India in 2013. His paper was published in their proceeding with ISBN