Identifying Nearly Duplicate Records In Relational Database

*Bhagyashri. A. Kelkar**
*ME (CSE) Research Scholar, D. Y. Patil college of Engg. & Tech., Kolhapur, Maharashtra, India.

*Prof. K. B. Manwade***
** Asst. Prof. Dept. of CSE, Annasaheb Dange college of Engg. & Tech.,Ashta,District Sangli, Maharashtra, India.

***Abstract*—Entity resolution is an important precess for many database based applications. Accurately identifying duplicate records between multiple data sources is a persistent problem that is big challenge to organizations and researchers. The aim of this process is to detect the approximately duplicate records that refer to the same real-world entity to make the database more concrete and achieve higher data quality. Though ideally each record must be compared with every other record in dataset for finding duplicates, it is possible to reduce search space for record comparisons by using mutual exclusion property of tuples. In this research paper we analyze two types of blocking algorithms, namely, the adaptive sorted neighborhood method (SNM), and iterative blocking and their combination with Jaro Winkler distance and Soundex phonetic algorithm for string matching. Experimental evaluation on real dataset shows that, Jaro Winkler distance algorithm outperforms in terms of precision and recall, whereas adaptive sorted neighborhood method requires very less number of comparisons than iterative blocking method. The experiments also highlight that, strings matching threshold of 85% gives optimal results in terms of precision, recall and F-measure.**

Keywords: Record linkage, near duplicate detection, Iterative Blocking, Adaptive sorted neighborhood method(ASNM), Jaro Winkler Distance, Soundex.

I. INTRODUCTION

Databases play an important role in today's IT based economy. Often there is need to consolidate data from different sources. The data integrated from the outside often involves a certain degree of duplicate records and the rate commonly varies between %1-5%. We say two records as nearly duplicate if they identify the same real world entity. Many industries and systems depend on the accuracy of databases for taking their operational, strategic and competitive initiatives. Quality of the information stored in the databases, can have significant cost implications to a system that relies on information to function and conduct business. Low data quality results in incorrect reporting, inability to create a complete view of the customers from various segments and also results in poor customer service and costs billions of dollars to businesses in postage, printing, and staff overhead. Hence, data quality improvement is an on going exercise and essential step before establishing data warehouse.

Presence of nearly duplicate records is result of expressing an entity using different values due to abbreviation, type errors, inconsistent expression habit & different formats. Thus, data quality is often compromised by many factors, including data entry errors (e.g., Microsft instead of Microsoft), missing integrity constraints (e.g., allowing entries such as EmployeeAge = 234), and multiple conventions for recording information (i.e. lexical heterogeneity) e.g., address recorded as “44 W. 4th St.” versus “44 West Fourth Street”.

The most naive method for finding nearly duplicate records is to compare all records in the database, pair wise. Obviously, this method is practically infeasible due to intolerable complexity of $O(n^2)$. To lower the time complexity, various techniques have been proposed.

II. REDUCE SEARCH SPACE WITH BLOCHING

Blocking refers to the procedure of subdividing database records into a set of mutually exclusive subsets (blocks) under the assumption that no matches occur across different blocks. In this paper, we analyze performance of iterative blocking and adaptive sorted neighborhood method.

A. Iterative Blocking:

Most blocking techniques process blocks separately and do not exploit the results of other blocks. In iterative blocking, results of blocks are reflected to subsequently processed blocks. Blocks are now iteratively processed until no block contains any more matching records. When two records match and merge in one block, their composite may match with records in other blocks. The same pair of records may occur in multiple blocks, so once the pair is compared in one block, we can avoid comparing it in other blocks.

B. Adaptive Sorted Neighborhood Method:

In this blocking method, blocking key values adjacent to each other, but that are significantly different from each other are found using an appropriate string similarity measure. These boundary pairs of blocking keys are then used to form blocks, i.e. they mark the positions in the sorted array where one window ends and a new one starts. This approach can therefore be seen as a combination of traditional blocking and the sorted neighborhood approach. Record pairs that were removed in the indexing step are classified as non-matches without being compared explicitly.

III. FIELD SIMILARITY COMPUTATION

Approach to reduce the computational efforts is to minimize the number of costly string comparisons that need to be made between records. Approximate string comparisons are carried out by using methods like edit distance, Jaccard similarity, Cosine similarity, matching tree etc. Jaro-Wrinkler algorithm is found effective for fields like name and address details, while comparison functions specific for date, age, and numerical values are used for fields that contain such data. In the proposed system, performance of Jaro-Winkler metric and Soundex - phonetic similarity metric will be compared on experimental dataset.

1. Attribute selection:

A record is usually composed of many attributes, whose similarities decide record similarity. It is vital to select the attributes of the records that participate in the record match, which represent the whole record. Attributes are selected by domain experts depending on people's comprehension of the meanings of data. If the cardinality of an attribute is approximately equal to the cardinality of dataset, then that attribute could not be a duplicate identifier. For example, title is not a duplicate identifier for name database.

2. Attribute weight allocation:

A record is usually composed of many attributes, whose contributions are different in deciding whether two pieces of records are approximate. Attribute weight is allotted according to the importance of its contribution in the process of judging approximately duplicate record. The bigger contribution the attribute makes, the bigger weight need to be allotted. In this paper, effect of attribute weights on precision and recall and F-measure is analyzed.

IV. FIELD SIMILARITY COMPUTATION

Field matching step results into formation of a vector that contains the numerical similarity values (F_i) calculated for each pair for selected attributes. Record similarity (RS) between two records R_1 & R_2 is

$$RS(R_1, R_2) = \frac{\sum_{i=1}^{n} F_i * W_i}{\sum_{i=1}^{n} W_i}$$

Where, W_i is weight allocated to i^{th} field and n Fields contribute to form the vector. If $RS(R_1, R_2)$ is greater than user specified threshold, record R_1 is marked as similar to R_2. This stage classifies the compared candidate record pairs into matches & non-matches. IN this paper, effect of threshold value on precision and recall, F-measure and number of record comparisons is analyzed.

V. EXPERIMENTAL RESULTS

The experiments are done on real world Restaurant dataset. Restaurant is a standard dataset which is used in several record linkage studies. It was created by merging the information of some restaurants from two websites: Zagat (331 non-duplicate restaurants) and Fooders (533 non-duplicate restaurants). There are 864 records in this dataset and 112 of them are duplicates. Name, Address, City, Phone and Type of restaurants are attributes of this dataset. Every record has five fields of the following form

Record (RIDDLE, Restaurant data set):@data
"arnie morton’s of chicago", "435 s. la cienega blv.", "los angeles", "310/246-1501", "american", ’0’
"arnie morton’s of chicago", "435 s. la cienega blvd.", "los angeles", "310-246-1501", "steakhouses", ’0’

Evaluation metrics: The effectiveness of the blocking methods and string similarity algorithms can be measured by precision, recall and F-measure metrics.

True Positive (TP): Corresponds to the number of matched detected when it is really match.

True Negative(TN): Corresponds to the number of non-matches detected when it is really non-match.

False Positive (FP): Corresponds to the number of matches detected when it is really non-match.

False Negative(FN): Corresponds to the numbers of non-matches detected when it is really match.

Precision: Precision is the fraction of true matches over the all number of candidate pairs which are classified as matches.

Recall: Recall is the fraction of matches correctly classified over the all number of matches.

F-measure: F-measure is regarded as the mean of precision and recall values.

Pairwise comparison count: This metric measures the number of pairwise comparisons performed by the algorithm. The lower the count, the more efficient the algorithm is.

Implementation of iterative blocking using Jaro Winkler string similarity:

Blocks were formed on city in first pass and then on telephone number in second pass. Similarity threshold values were varied from 70% to 91%. All the contributing fields i.e. name, address, city, phone and cuisine type were assigned equal weights while calculating record similarity index. The results of Precision, Recall, F-measure & number of comparisons required are as shown following table.

TABLE I: RESULTS OF ITERATIVE BLOCKING WITH JARO-WINKLER SIMILARITY AND EQUAL FIELD WEIGHTS

Threshold %	**Precision**	**Recall**	**F-measure**	**#Comparisons**
70	0.75	0.88	0.81	59395
80	0.87	0.88	0.87	61370
84	0.96	0.88	0.92	62994
85	0.98	0.88	0.93	63490
86	0.98	0.86	0.92	63602
90	0.99	0.73	0.84	66174
91	0.98	0.72	0.83	66262

Above results show that, at 85% of similarity threshold, F-measure, precision, recall are at optimal level. At 90% threshold, precision maximize but recall and F-measure drops as more number of records are rejected from comparisons. With increase in threshold value, number of record comparisons also increase.

If weights of fields were kept as name(30%), address(30%), city(20%) , cuisine_type(20%), then following are the results of F-measure, precision, recall & number of comparisons required.

TABLE II:RESULTS OF ITERATIVE BLOCKING USING JARO-WINKLER SIMILARITY AND UNEQUAL FIELD WEIGHTS

Threshold %	Precision	Recall	F-measure	#Comparisons
70	0.74	0.80	0.77	60620
80	0.86	0.82	0.84	62578
84	0.96	0.80	0.87	64904
85	0.98	0.77	0.86	66129
86	0.98	0.75	0.85	66544
90	0.99	0.62	0.76	68305
91	0.97	0.60	0.74	68486

Comparison of table I with table II indicate that, assigning unequal weights for forming similarity index results in reduction in precision, recall & F-measure and increase in number of record comparisons.

Implementation of iterative blocking & Soundex codes:

Soundex code is generated for each field being compared, i.e. name, address, city and cuisine type. For approximate matching, Soundex codes can not be matched exactly. If the similarity index of Soundex codes of fields under comparison is greater than threshold value, the fields are marked as matching and records similarity index is increased by value of weight assigned for that field. This process continues for all fields contributing for record similarity index calculation. If record similarity >= 0.75 the records are marked as matching. Threshold values were varied from 30% to 95%. The results of Precision, Recall, F-measure & number of comparisons required are as shown following table.

TABLE III. RESULTS OF ITERATIVE BLOCKING WITH SOUNDEX SIMILARITY AND EQUAL FIELD WEIGHTS

Threshold %	Precision	Recall	F-measure	#Comparisons
30	0.93	0.85	0.89	62907
40	0.97	0.82	0.89	64017
50	0.96	0.68	0.80	66646
60	0.97	0.66	0.79	66987
70	0.97	0.66	0.79	66987
80	0.99	0.64	0.78	67390
85	0.98	0.58	0.73	68874
90	0.98	0.56	0.71	69050
95	0.98	0.56	0.71	69050

Above results show that with Soundex similarity function, at 40% of similarity threshold, F-measure, precision, recall maximize. Thereafter with increase in threshold value, recall & precision drops, precision remains nearly constant and number of record comparisons increase.

Implementation of adaptive sorted neighborhood with Jaro-Winkler string similarity:

The data was then sorted on sort key 1 in first pass and on sort key 2 in second pass. For the first pass, sort key was created by concatenating first two characters of each of following fields : name, address, city, phone and cuisine type in order. In the second pass sort key was created by concatenating first two characters of fields city, phone, cuisine type, name and address in sequence.

A fixed window size is defined initially and changed adaptively when a match is found. Comparisons of records take place within the window only. First record in result set is selected as base record and compared with remaining records one after the other within the window. Jaro-Winkler string similarity is used to find mathcing index of the records. If a matching record is found, i.e. match index > threshold value, remaining records in the window were bypassed and next record of current base record is marked as new base record. If the new record is already matched with some other record, it need not be processed further and so bypassed and next base record is selected for processing. This process is continued till all records exhaust. The results of Precision, Recall, F-measure and number of comparisons required are as shown in following table.

TABLE RESULTS OF ADAPTIVE SORTED NEIGHBORHOOD METHOD WITH JARO-WINKLER SIMILARITY AND EQUAL FIELD WEIGHTS

Threshold %	Precision	Recall	F-measure	#Comparisons
70	0.79	0.87	0.83	4258
80	0.85	0.88	0.86	4307
84	0.93	0.88	0.90	4382
85	0.96	0.88	0.92	4406
86	0.97	0.86	0.91	4430
90	0.97	0.76	0.85	4520
91	0.98	0.74	0.84	4544

Above results show that, at 85% of similarity threshold, F-measure, precision, recall reach optimum values. At 91% threshold, precision maximizes but recall and F-measure drop as more number of records are rejected from comparisons. Adaptive approach results in tremendous drop in number of comparisons.

Implementation of adaptive sorted neighborhood with Soundex string similarity:

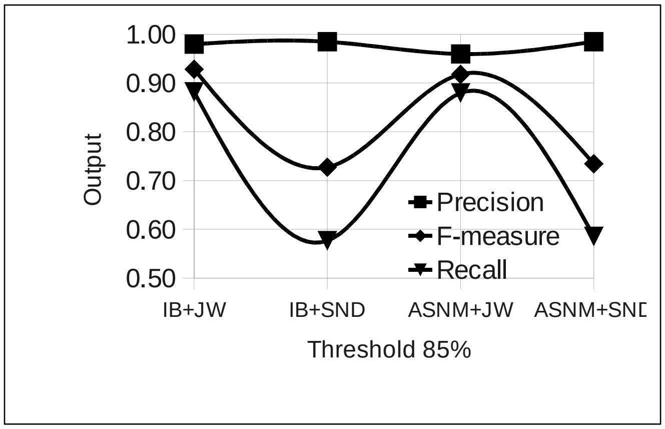
For field similarity computation Soundex codes were used as explained in iterative blocking with Soundex similarity framework above. The results of Precision, Recall, F-measure and number of comparisons required are as shown in table V. It is clear that with Soundex similarity function at 40% similarity threshold, F-measure, precision, recall are at optimum value. Thereafter with increase in threshold value, recall & precision drop & precision remains nearly constant.

Following graph clearly indicates that adaptive sorted neighborhood method requires very few record comparisons than iterative blocking method.

TABLE V. RESULTS OF ADAPTIVE SORTED NEIGHBORHOOD METHOD WITH SOUNDEX SIMILARITY AND EQUAL FIELD WEIGHTS

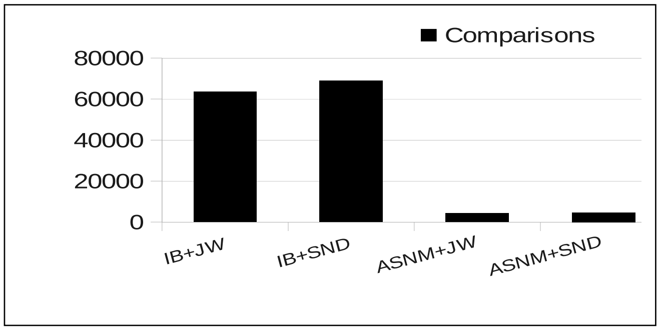
Threshold %	Precision	Recall	F-measure	#Comparisons
30	0.91	0.86	0.89	4390
40	0.96	0.83	0.89	4449
50	0.95	0.69	0.80	4568
60	0.97	0.67	0.80	4586
70	0.97	0.67	0.80	4586
80	0.99	0.65	0.78	4620
85	0.98	0.59	0.73	4685
90	0.98	0.57	0.72	4692
95	0.98	0.57	0.72	4692

Figure 1. Comparative chart of iterative blocking(IB) and adaptive sorted neighborhood(ASNM) in combination with Jaro-Winkler(JW) and Soundex(SND) similarity



Results obtained by combination of iterative blocking and adaptive sorted neighborhood method with Jaro Winkler and Soundex at threshold value of 0.85 & equal weight assigned to contributing fields are shown in Fig. 1. It shows that Jaro-Winkler string similarity works well than Soundex for maximizing precision, recall and F-measure. Fig. 2 shows that, adaptive sorted neighborhood method reduces number of record comparisons by factor of 93%

Figure 2. Number of comparison required for iterative blocking(IB) and adaptive sorted neighborhood(ASNM) in combination with Jaro-Winkler(JW) and Soundex(SND) similarity



VI. CONCLUSION

The results show that, similarity threshold should not be kept more than 90% because recall and F-measure fall as true positives do not match at this threshold. Below 70% threshold, precision falls as false positives are increased. Irrespective of blocking method, Jaro-Winkler string matching algorithm gives optimum results at threshold of 85% with average number of record comparisons. Equal weight combination requires less comparisons than that for any other weight combination. Sorted neighborhood method reduces number of comparisons by factor of 93% at cost of 0.2% of precision. The results also highlight that, combination of adaptive sorted neighborhood method and Jaro Winkler similarity outperforms remaining combinations. Iterative blocking is suitable for applications requiring high precision and small to moderate sized databases, whereas adaptive sorted neighborhood method is suitable for large databases. Finding other methods to enhance the effectiveness of detecting duplicate records, such as combining similarity measures in classification or finding more proper similarity measures is future research area.

REFERENCES

[1] A. E. Monge and C. Elkan, “An efficient domain-independent algorithm for detecting approximately S.E. Whang, D. Menestrina, G. Koutrika, M. Theobald, H. Garcia-Molina, Entity resolution with iterative blocking, in: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD ’09), 2009, pp. 219–232.

[2] Xiaochun Yang Bin Wang Guoren Wang Ge Yu Key “RSEARCH: Enhancing Keyword Search in Relational Databases Using Nearly Duplicate Records” (2010 IEEE. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering)

[3] Ahmed K. Elmagarmid, Senior Member, IEEE, Panagiotis G. Ipeirotis, Member, IEEE Computer Society, and Vassilios S. Verykios, Member, IEEE Computer Society Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S.Verykios. “Duplicate Record Detection: A Survey”(IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 19, NO. 1, JANUARY 2007)

[4] Su Yan, Dongwon Lee, Min-Yen Kan, and Lee C. Giles. “Adaptive sorted neighborhood methods for efficient record linkage.” In Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries, 2007.

[5] Steven Euijong Whang, David Menestrina, Georgia Koutrika, Martin Theobald, and Hector Garcia-Molina. “Entity resolution with iterative blocking”. In Proceedings of the ACM International Conference on Management of Data (SIGMOD), 2009.

[6] Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. “Adaptive name matching in information integration” IEEE Intelligent Systems, 18(5):16-23, Sep/Oct 2003.

[7] AUTHORS PROFILE

Bhagyashri A. Kelkar received the BE in computer sci. & engg. from Walchand College of Engg., Sangli in 1995. She has 13 years of experience in database based application s/w development. She is working as lecturer in D.Y Patil College of engg. & tech., Kolhapur and is currently pursuing ME degree in computer sci. & engg.

K. B. Manwade received M.Tech. degree in computer science from Shivaji University, Kolhapur. He is working as an assistant professor in Annasaheb Dange College of engg. & tech., Ashta , Sangli.