Investigation of Techniques for Efficient & Accurate Indexing for Scalable Record Linkage & Deduplication

Sunitha Yeddula, K.Lakshmaiah

Abstract- Record linkage is the process of matching records from several databases that refer to the same entities. When applied on a single database, this process is known as deduplication. Increasingly, matched data are becoming important in many applications areas, because they can contain information that is not available otherwise, or that is too costly to acquire. Removing duplicate records in a single database is a crucial step in the data cleaning process. and also, the complexity of the matching process becomes one of the major challenge. Various indexing techniques have been developed for record linkage and deduplication. They are aimed at reducing the number of record pairs to be compared in the matching process by removing obvious non-matching pairs, while at the same time maintaining high matching quality. This paper presents a survey of variations of six indexing techniques. Their complexity is analyzed, and their performance and scalability is evaluated within an experimental framework using both synthetic and real data sets.

Keywords - Data matching, data linkage, entity resolution, index techniques, blocking, experimental evaluation, scalability.

I. INTRODUCTION

AS many businesses, government agencies and research projects collect increasingly large amounts of data, techniques that allow efficient processing, analyzing and mining of such massive databases have in recent years attracted interest from both academia and industry. One task that has been recognized to be of increasing importance in many application domains is the matching of records that relate to the same entities from several databases. Often, information from multiple sources needs to be integrated and combined in order to improve data quality, or to enrich data to facilitate more detailed data analysis.

The task of record linkage is now commonly used for improving data quality and integrity, to allow re-use of existing data sources for new studies, and to reduce costs and efforts in data acquisition. In the health sector, for example, matched data can contain information that is required to improve health policies, information that traditionally has been collected with time consuming and expensive survey methods[5],[6]. Linked data can also help in health surveillance systems to enrich data that is used for the detection of suspicious patterns Statistical agencies have employed record linkage for several decades on a routinely basis to link census data for further analysis. Many businesses use deduplication and record linkage techniques with the aim to deduplicate their databases to improve data quality or compile mailing lists, or to match their data across organizations.

Manuscript received September 02, 2012.

Sunitha Yeddula, (M.Tech),IInd yr,cse, MITS, Madanapalle, Chittoor dist, A.P, India,

K.Lakshmaiah, M.tech.,(Ph.D),Associate Profes- or, cse dept, MITS, Madanapalle, Chittoor dist, A.P, India,

The problem of finding records that relate to the same entities not only applies to databases that contain information about people. In the field or information retrieval, it is important to remove duplicate documents in the results returned by search engines, in digital libraries or in automatic text indexing systems [7],[8]. Another application of growing interest is finding and comparing consumer products from different online stores.

In situations where unique entity identifiers (or keys) are available across all the databases to be linked, the problem of matching records at the entity level becomes trivial: a simple database join is all that is required. However, in most cases no such unique identifiers are shared by all databases, and more sophisticated linkage techniques are required. In commercial processing of business mailing lists and customer databases, record linkage is usually seen as a component of *ETL* (extraction, transformation and loading) tools.

The indexing step generates candidate record pairs, while the outputs of the comparison step are vectors containing numerical similarity values.

Figure 1 outlines the general steps involved in the linking of two databases. Because most real-world data are dirty and contain noisy, incomplete and incorrectly formatted information,

First step: in any record linkage or deduplication project is data cleaning and standardization [1].

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.

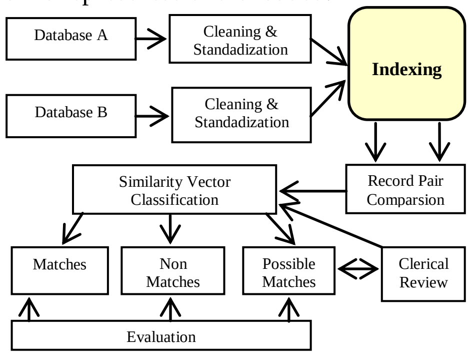

Figure 1. Outline of the general record linkage process

Second step: ('Indexing') is the topic of this survey, in which the indexing step generates pairs of candidate records.

These records are compared in detail in the comparison step using a variety of comparison functions appropriate to the content of the record fields (attributes). The next step in the record linkage process is to classify the compared candidate record pairs into matches, non-matches, and possible matches, depending upon the decision model used. If record pairs are classified into possible matches, a clerical review process is required where these pairs are manually assessed and classified into matches or no matches. Measuring and evaluating the quality and complexity of a record linkage project is a final step in the record linkage process.

II. INDEXING FOR RECORD LINKAGE & DEDUPLICATION

When two databases, A and B, are to be matched, potentially each record from A needs to be compared with every record from B, resulting in a maximum number of |A| × |B| comparisons between two records. Similarly, when deduplicating a singe database A, the maximum number of possible comparisons is |A|× (|A|− 1)/2, because each record in A potentially needs to be compared with all other records. The performance bottleneck in a record linkage or deduplication system is usually the expensive detailed comparison of field (attribute) values between records, making the naive approach of comparing all pairs of records not feasible when the databases are large.

At the same time, assuming there are no duplicate records in the databases to be matched, then the maximum possible number of true matches will correspond to min(|**A**|, |**B**|). Similarly, for a deduplication the number of unique entities in a database is always smaller than or equal to the number of records in it. Therefore, while the computational efforts of comparing records increase quadratically as databases are getting larger, the number of potential true matches only increases linearly in the size of the databases.

It is clear that the vast majority of comparisons will be between records that are not matches. The aim of the indexing step is to reduce this large number of potential comparisons by removing as many record pairs as possible that correspond to non matches. The traditional record linkage approach[3],[4] has employed an indexing technique commonly called *blocking[2]*, which splits the databases into non-overlapping blocks, such that only records within each block are compared with each other. A blocking criterion, commonly called a *blocking key*, is either based on a single record field (attribute), or the concatenation of values from several fields. an important criteria for a good blocking key is that it can group similar values into the same block. What constitutes a 'similar' value depends upon the characteristics of the data to be matched. Similarity can refer to similar sounding or similar looking values based on phonetic or character shape characteristics.

Several important issues need to be considered when record fields are selected to be used as blocking keys.

The first issue is that the quality of the values in these fields will influence the quality of the generated candidate record pairs. Ideally, fields containing the fewest errors, variations or missing values should be chosen. Any error in a field value used to generate a BKV will potentially result in records being inserted into the wrong block, thus leading to missing true matches.

A second issue that needs to be considered when defining blocking keys is that the frequency distribution of the values in the fields used for blocking keys will affect the size of the generated blocks. Often this will be the case even after phonetic or other encodings have been applied. The largest blocks generated in the indexing step will dominate execution time of the comparison step, because they will contribute a large portion of the total number of candidate record pairs. Therefore, it is of advantage to use fields that contain uniformly distributed values because they will result in blocks of equal sizes.

When blocking keys are defined, there is also a tradeoff that needs to be considered. On one hand, having a large number of smaller blocks will result in fewer candidate record pairs that will be generated. This will likely increase the number of true matches that are missed. On the other hand, blocking keys that result in larger blocks will generate an increased number of candidate record pairs that likely will cover more true matches, at the cost of having to compare more candidate pairs.

III. INDEXING TECHNIQUES

When two databases In this section, the traditional blocking approach and five more recently developed indexing techniques and variations of them are discussed in more detail.

The estimated number of candidate record pairs will be calculated for two different frequency distributions of BKVs. The first assumes a uniform distribution of values, resulting in each block containing the same number of records. The second assumes that the frequencies of the BKVs follow Zipf's law[9], a frequency distribution that is commonly found in data sets that contain values such as personal names. Zipf's law states that in a list of words ranked according to their frequencies. Conceptually, the indexing step of the record linkage process can be split into the following two phases:

1) Build - All records in the database (or databases) are read, their BKVs (blocking key value) are generated, and records are inserted into appropriate index data structures. For most indexing techniques, an inverted index can be used. The BKVs will become the keys of the inverted index, and the record identifiers of all records that have the same BKV will be inserted into the same inverted index list. Figure 2 illustrates this for a small example data set.

Identifiers	Surnames	BKVs (Soundex encoding)
R1	Smith	S530
R2	Miller	M460
R3	Peters	P362
R4	Myler	M460
R5	Smyth	S530
R6	Millar	M460
R7	Smyth	S530
R8	Miller	M460

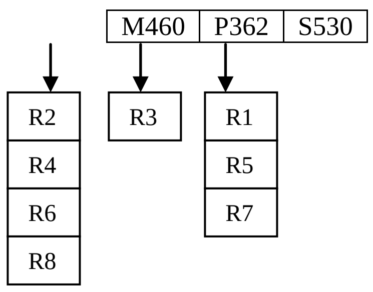

Figure 2. Example records with surname values and their soundex encodings used as blocking key values, and the corresponding inverted index data structure as used for traditional blocking.

When linking two databases, either a separate index data structure is built for each database, or a single data structure with common key values is generated. For the second case, each record identifier needs to include a flag that indicates from which database the record originates.

2) Retrieve - For each block, its list of record identifiers is retrieved from the inverted index, and candidate record pairs are generated from this list. For a record linkage, all records in a block from one database will be paired with all records from the block with the same BKV from the other database, while for a deduplication each record in a block will be paired with all other records in the same block. For example, from the block with key 'S530' from Figure2 the pairs (R1, R5), (R1, R7) and (R5, R7) will be generated.

A. Traditional Blocking

This technique has been used in record linkage since the 1960s [4]. All records that have the same BKV are inserted into the same block, and only records within the same block are then compared with each other. Each record is inserted into one block only (assuming a single blocking key definition). Traditional blocking can be implemented efficiently using a standard inverted index [9], the identifiers of all records in the same block are retrieved and the corresponding candidate record pairs are generated. While traditional blocking does not have any explicit parameters, the way blocking keys are defined will influence the quality and number of candidate record pairs that are generated.

A major drawback of traditional blocking is that errors and variations in the record fields used to generate BKVs will lead to records being inserted into the wrong block. This drawback can be overcome by using several blocking key definitions based on different record fields, or different encodings applied on the same record fields. A second drawback of traditional blocking is that the sizes of the blocks generated depend upon the frequency distribution of the BKVs, and thus it is difficult in practice to predict the total number of candidate record pairs that will be generated.

B. Sorted Neighbourhood Indexing

This technique was first proposed in the mid 1990s [10]. Its basic idea is to sort the database(s) according to the BKVs, and to sequentially move a window of a fixed number of records w (w > 1) over the sorted values. Candidate record pairs are then generated only from records within a current window.

1) Sorted Array Based Approach

In this first approach, as originally proposed, the BKVs are inserted into an array that is sorted alphabetically. The window is then moved over this sorted array and candidate record pairs are generated from all records in the current window. In case of a record linkage, the BKVs from both databases will be inserted into one combined array and then sorted alphabetically, but candidate record pairs are generated in such a way that for each pair one record is selected from each of the two databases.

2) Inverted Index Based Approach

It is an alternative approach [11] for the sorted neighborhood. Rather than inserting BKVs into a sorted array, this approach utilizes an inverted index similar to traditional blocking. The index keys contain the alphabetically sorted BKVs, the window is moved over these sorted BKVs, and candidate record pairs are formed from all records in the corresponding index lists. Similar to the sorted array based approach, most candidate record pairs are generated in several windows, but each unique candidate pair will again only be compared once in the comparison step. The number of generated candidate record pairs with this approach depends upon the number of record identifiers that are stored in the inverted index lists.

3) Adaptive Sorted Neighbourhood Approach

Recent research has looked at how the sorted neighbourhood indexing technique based on a sorted array can be improved. The issue of having a fixed block size w which can result in missed true matches (because not all same BKVs fit into one window) has been addressed through an adaptive approach to dynamically set the window size. Due to the adaptive nature of the approach, where block sizes are determined by the similarities between BKVs.

C. Q-gram Based Indexing

This technique aims to allow for 'fuzzy' blocking, by converting the blocking key values into lists of q-grams (sub-strings of length q), and, based on sub-lists of these q-gram lists, each record is inserted into several blocks according to a Jaccard - based similarity threshold. While this technique improves entity resolution for data that contains a large proportion of errors and modifications, its computational complexity makes it unsuitable for large databases.

D. Suffix Array Based Indexing

The basic idea of this suffix array based indexing technique [12] is to insert the blocking key values and their suffixes into a suffix array based inverted index. A suffix array contains strings or sequences and their suffixes in an alphabetically sorted order. Similar to canopy clustering, each record might be inserted into several blocks, depending upon the length of their blocking key values. Record pairs will then be formed from all pairs that are in the same inverted index list.

1) Robust Suffix Array Based Indexing

An improvement upon the original suffix array based indexing technique has recently been proposed. Similar to adaptive blocking, the inverted index lists of suffix values that are similar to each other in the sorted suffix array are merged. An approximate string similarity measure is calculated for all pairs of neighboring suffix values, and if the similarity of a pair is above a selected threshold t, then their lists are merged to form a new larger block.

E. Canopy Clustering

This technique is based on the idea of using a computationally cheap clustering approach to create high-dimensional overlapping clusters, from which blocks of candidate record pairs can then be generated[13],[14]. Clusters are created by calculating the similarities between BKVs using measures such as Jaccard or TF-IDF/cosine. Both of these measures are based on tokens, which can be characters, qgrams or words. They can be implemented efficiently using an inverted index which has tokens, rather than the actual BKVs, as index keys.

1) Threshold Based Approach

In this originally proposed approach [13],[14], two similarity thresholds are used to create the overlapping clusters. All records rx that are within a loose similarity, tl, to rc are inserted into the current cluster (e.g. all records with tl ≤ sJ). Of these, all records that are within a tight similarity threshold tt (with tt ≥ tl), will be removed from the pool of candidate records. This process of randomly selecting a centroid record rc, calculating the similarities between this and all other records in the pool, and inserting records into clusters, is repeated until no candidate records are left in the pool. If tl = tt, the clusters will not be overlapping, which means each record will be inserted into one cluster only. If both tl = 1 and tt = 1 (i.e. exact similarity only), canopy clustering will generate the same candidate record pairs as traditional blocking.

2) *Nearest Neighbour Based Approach*

An alternative to using two thresholds is to employ a nearest neighbor based approach to create the overlapping clusters. The idea is to replace the two threshold parameters, tl and tt, with two nearest neighbour parameters, nl and nt (with nl ≥ nt). The first parameter, nl, corresponds to the number of record identifiers that are inserted into each cluster, while nt is the number of record identifiers that are removed from the pool of candidate records in each step of the algorithm. Similar to the threshold based approach, the process of creating overlapping clusters starts by randomly selecting a record rc from the pool of initially all records. Similarities are then calculated between the rc and the records rx that have tokens in common in the inverted index. The nl records closest to rc are inserted into the current cluster, and of these the nt records closest to rc are removed from the pool.

This approach will result in all clusters containing nl record identifiers, independently of the frequency distribution of the BKVs. Therefore, blocks of uniform size will be created, allowing the calculation of the number of generated record pairs. The number of clusters only depends upon the number of records in the database(s) to be matched or deduplicated, and the values of nl and nt. The number of clusters generated corresponds to nA/nt and nB/nt, respectively, and each cluster will contain nl records.

F. String-Map Based Indexing

This indexing technique [15] is based on mapping BKVs to objects in a multi-dimensional euclidean space, such that the distances between pairs of strings are preserved. Any string similarity measure that is a distance function can be used in the mapping process. Groups of similar strings are then generated by extracting objects in this space that are similar to each other. The approach is based on a modification of the *FastMap* algorithm, called *StringMap* that has a linear complexity in the number of strings to be mapped. Similar to canopy clustering based indexing; overlapping clusters can be extracted from the multidimensional grid index. An object (referring to a BKV) is randomly picked from the pool of (initially all) objects in the grid based index, and the objects in the same, as well as in neighbouring grid cells, are retrieved from the index. Similar to canopy clustering, either two thresholds, tl and tt, or the number of nearest neighbours, nl and nt, can be used to insert similar objects into clusters, and remove objects from the pool with a similarity larger than tt, or that are the nt nearest objects to the centroid object.

IV. EXPERIMENTAL EVALUATIONS

The aim of the experiments conducted was to evaluate the presented indexing techniques within a common framework, to answer questions such as: How do parameter values and the choice of the blocking key influence the number and quality of the candidate record pairs generated? How do indexing techniques perform with different types of data? Which indexing techniques show better scalability to larger databases?

A. Test Data Sets

Two series of experiments were conducted, the first using four ‘real’ data sets that have previously been used by the record linkage research community, and the second using artificial data sets. Table 3 summarizes these data sets. The aim of the first series of experiments was to investigate how different indexing techniques are able to handle various types of data, while the second series was aimed at investigating the scalability of the different indexing techniques to larger data sets. The first three ‘real’ data sets were taken from the *Second String* toolkit1. ‘Census’ contains records that were generated by the US Census Bureau based on real census data; ‘Cora’ contains bibliographic records of machine learning publications; and ‘Restaurant’ contains records extracted from the Fodor and Zagat restaurant guides. The ‘CDDB’ data set contains records of audio CDs, such as their title, artist, genre and year. The true match status of all record pairs is available in all four data sets.

Artificial data sets were generated using the *Febrl* data generator. This generator first creates *original* records based on frequency tables that contain real name and address values, as well as other personal attributes; followed by the generation of duplicates of these records based on random modifications.

In Table 1, the data sets used in experiment are artificial data sets containing 1000, 5000, 10000, 50000 and 100000 records, respectively, were generated.

Table 1

Data sets used in experiments.

Data set name	**Task**	**Number of records**	**Total number of true matches**
Census	Linkage	544+479	390
Restaurant	Deduplication	820	108
Cora	Deduplication	1560	19145
CDDB	Deduplication	9763	607
Clean	Linkage	1000-100000	200-20000
Dirty	Linkage	1000-100000	400-40000

As shown in Table, two series of artificial data sets were created. The ‘Clean’ data contain 80% original and 20% duplicate records, with up to three duplicates for one original record, a maximum of one modification per attribute, and a maximum of three modifications per record. The ‘Dirty’ data contain 60% original and 40% duplicate records, with up to nine duplicates per original record, a maximum of three modifications per attribute, and a maximum of ten modifications per record.

B. Quality and Complexity Measures

Four measures are used to assess the complexity of the indexing step and the quality of the resulting candidate record pairs [9], [10]. The total number of matched and non-matched record pairs are denoted with nM and nN, respectively, with nM + nM = nA × nB for the linkage of two databases, and nM + nN = nA(nA − 1)/2 for the deduplication of one database. The number of true matched and true non-matched candidate record pairs generated by an indexing technique is denoted with sM and sN, respectively, with sM + sN ≤ nM + nN.

The reduction ratio, RR = 1.0− (sM+sN) / (nM+nN), measures the reduction of the comparison space, i.e. the fraction of record pairs that are removed by an indexing technique. The higher the RR value, the less candidate record pairs are being generated.

Table 2

The label used in the result figures, the number of different parameter setting evaluated, and the run-times in milli-seconds per candidate pair required to build each of the evaluated indexing techniques.

Indexing techniques	Label used in figures	No. of settings	**Time in milli-seconds per candidate record pair**			
			Minimum	Median	Average	**Maximum**
Traditional blocking	TB1o	1	0.002	0.585	0.521	**0.986**
Sorted neighborhood						
Array based	SorAr	5	0.011	0.059	0.081	**0.352**
Inverted Index	SorII	5	0.002	0.031	0.275	**3.032**
Adaptive	AdSor	8	0.002	0.952	1.228	**6.721**
Q-Gram						
Q-gram based indexing	QGr	4	0.005	2.118	1270.716	**193454.611**
Canopy clustering						
Threshold	CaTh	8	0.003	4.252	38.194	**440.265**
Nearest Neighborhood	CaNN	8	0.005	1.045	1.912	**19.127**
String map						
Threshold	STMTh	32	0.004	0.488	21.715	**466.890**
Nearest Neighborhood	StMNN	32	0.018	2.033	25.254	**392.322**
Suffix Array						
Suffix Array	SuAr	6	0.024	1.115	12.493	**264.967**
Suffix Array With Sub Strings	SuArSu	6	0.015	2.542	18.188	**338.102**
Robust	**RoSuA**	**48**	**0.009**	**0.334**	**0.456**	**12.588**

However, reduction ratio does not take the quality of the generated candidate record pairs into account (how many are true matches or not). Pairs completeness, PC = sM+nM , is the number of true matched candidate record pairs generated by an indexing technique divided by the total number of true matched pairs. Finally, pairs quality, PQ = sM/sM+sN , is the number of true matched candidate record pairs generated by an indexing technique divided by the total number of candidate pairs generated. A high PQ value means an indexing technique is efficient and generates mostly true matched candidate pairs.

V. CONCLUSION

The number of candidate record pairs generated by these techniques has been estimated and their efficiency and scalability has been evaluated using various data sets. These experiments highlight that one of the most important factors for efficient and accurate indexing for record linkage and deduplication is the proper definition of blocking keys. Because training data in the form of known true matches and non-matches is often not available in real world applications. The indexing techniques in this investigation are heuristic approaches that aim to split the records in a database into blocks such that matches are inserted in to the same block and non-matches in to different blocks.

ACKNOWLEDGMENT

Students work is incomplete until they thank the almighty & his teachers. I sincerely believe in this and would like to thank Mr.K. Lakshmaiah, M.Tech (Ph.D), Associate prof., of the department of Computer Science & Engineering, MITS Madanapalle, for his encouragement, motivation and for guiding me to write this paper. Also I am grateful to Dr.S.Murali Krishna, M.Tech., Ph.D. Prof., in CSE and Head Of the Department of Computer Science, MITS, Madanapalle .

REFERENCES

1. T. Churches, P. Christen, K. Lim, and J. X. Zhu, "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.
2. R. Baxter, P. Christen, and T. Churches, "A comparison of fast blocking methods for record linkage," in *ACM SIGKDD'03 workshop on Data Cleaning, Record Linkage and Object Consolidation*, Washington DC, 2003, pp. 25–27.
3. W. E. Winkler, "Overview of record linkage and current research directions," US Bureau of the Census, Tech. Rep. RR2006/02, 2006.
4. . P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Society*, vol. 64, no. 328, 1969.
5. D. E. Clark, "Practical introduction to record linkage for injury research," *Injury Prevention*, vol. 10, pp. 186–191, 2004.
6. C. W. Kelman, J. Bass, and D. Holman, "Research use of linked health data – A best practice protocol," *Aust NZ Journal of Public Health*, vol. 26, pp. 251–255, 2002
7. H. Hajishirzi, W. Yih, and A. Kolcz, "Adaptive near-duplicate detection via similarity learning," in *ACM SIGIR'10*, Geneva, Switzerland, 2010, pp. 419–426.
8. W. Su, J. Wang, and F. H. Lochovsky, "Record matching over query results from multiple web databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 4, pp. 578–589, 2009.
9. I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes*, 2nd ed. Morgan Kaufmann, 1999.
10. M. A. Hernandez and S. J. Stolfo,"The merge/purge problem for large databases," in *ACM SIGMOD'95*, San Jose, 1995.
11. P. Christen, "Towards parameter-free blocking for scalable record linkage, "Department of Computer Science, The Australian National University, Canberra, Tech. Rep. TR-CS-07-03, 2007.
12. A. Aizawa and K. Oyama, "A fast linkage detection scheme for multi-source information integration," in *WIRI'05*, Tokyo, 2005.
13. W. W. Cohen and J. Richman, "Learning to match and cluster large high-dimensional data sets for data integration," in *ACM SIGKDD'02*, Edmonton, 2002, pp. 475–480.
14. A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *ACM SIGKDD'00*, Boston, 2000, pp. 169–178.
15. L. Jin, C. Li, and S. Mehrotra, "Efficient record linkage in large data sets," in DASFAA'03, Tokyo, 2003, pp.137.