RESEARCH ARTICLE OPEN ACCESS

Complexity Anaylsis of Indexing Techniques for Scalable Deduplication and Data Linkage

Koneru Naga Venkata Rajeev, Vanguru Radha Krishna, Kambhampati Vamsi Krishna, Siddhavatam Thirumala Reddy
School of Information Technology and Engineering VIT UNIVERSITY Vellore, India
School of Information Technology and Engineering VIT UNIVERSITY Vellore, India
School of Information Technology and Engineering VIT UNIVERSITY Vellore, India
School of Information Technology and Engineering VIT UNIVERSITY Vellore, India

ABSTRACT
The process of record or data matching from various databases regarding same entities is called Data Linkage. Similarly, when this process is applied to a particular database, then it is called Deduplication. Record matching plays a major role in today's environment as it is more expensive to obtain. The process of data cleaning from database is the first and foremost step since copy of data severely affects the results of data mining. As the amount of databases increasing day-by-day, the difficulty in matching records happens to be a most important confront for data linkage. So many indexing techniques are designed for the process of data linkage. The main aim of those indexing techniques is to minimize the number of data pairs by eliminating apparent non-matching data pairs by maintaining maximum quality of matching. Hence in this paper, a survey is made of these indexing techniques to analyze complexity and evaluate scalability using fake data sets and original data sets.
Keywords: Complexity, Data matching, Indexing techniques.

I. Introduction

Today in real world one task is getting more significance in a number of fields. That is nothing but the data matching that connected to same objects from numerous databases. And this data from several sources should be integrated and unified to increase quality of data. Record linkage is employed in so many sectors including government agencies to recognize people who register for help or support multiple numbers of times. Even in the domain of detecting scams and crimes, this record linkage technique is useful. To access files for a specific person in enquiry or to cross check the histories of that person from multiple databases, record linkage plays a major role.

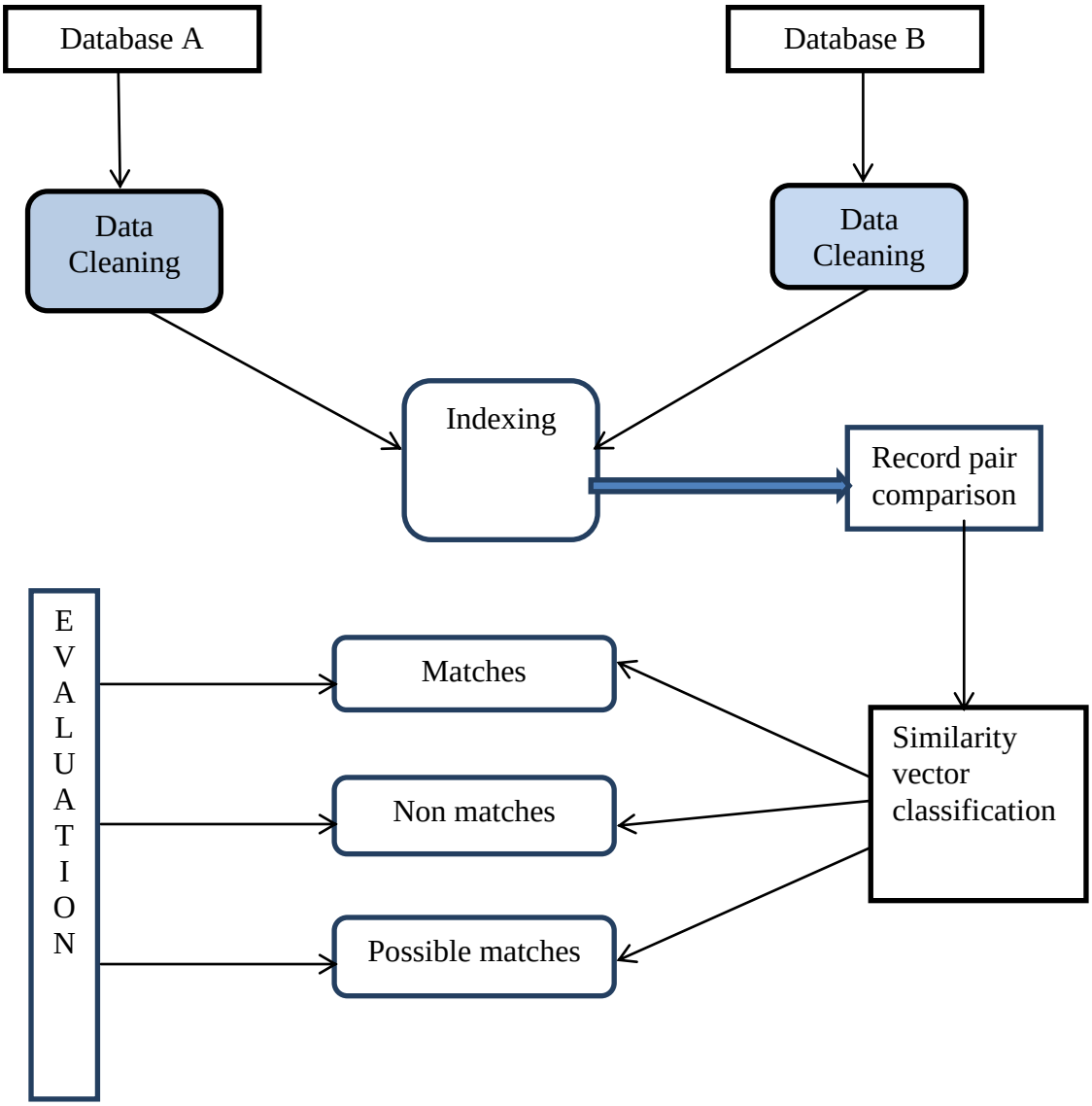

Fig. 1 Record Linkage Process

In biotechnology, record linkage is used to identify genome sequences in a huge number of data collections. In the area of information retrieval, it is significant to eliminate identical documents. The way of matching data is called data linkage by health researchers. But in other communities such as database and computer science, it is referred to field matching [1], duplicate detection [2], [3], information integration [4], data scrubbing or cleaning [5]. Data linkage is nothing but a component of ETL tools. Some recent surveys have delivered techniques and experiments regarding deduplication and linkage [3].

In Fig. 1, necessary steps needed to link two databases are described. As real-world data contain noisy data, inconsistent data, first it should be cleaned. Absence of good quality data leads to unsuccessful record linkage [6]. The next step is indexing, in which it creates sets of candidate records that will be distinguished in the next comparison step. After the comparison, based on their equality, it is divided into three types, which are possible matches, matches and non matches. Based on the view of possible matches, they are classified either into match or non-match. Finally, the complexity is evaluated or analyzed in the last step.

II. Indexing for Deduplication and Data Linkage

When different databases are to be compared, each and every record from one particular database should be matched with every record of the other database. Hence it leads to product i.e., $\mathbf{n(P) \times n(Q)}$ comparisons among the records where $\mathbf{n(.)}$ is number of records in a particular database. In the same way, if it is only for a single database then the number of comparisons

is $\mathbf{n(P) \times \frac{n(P) - 1}{2}}$. But the comparison of values is more expensive when it involves a large number of databases. If there are no matching records, then min $\mathbf{(n(P), n(Q))}$ is the number of comparisons.

Table.1 Examples for Generation of Blocking Keys

Record Fields					**Blocking Keys**		
Identifiers	Names(N)	Initials (I)	Codes (C)	Places(P)	Sndx(N)+C	Fi2D(C)+DME(I)	Sndx(P)+La2D(C)
A1	Rahul	Christen	1004	Chennai	R190-1004	**10-KRST**	C345-04
A2	Rania	Kristen	1011	Hyderabad	R190-1011	**10-KRST**	H470-11
A3	Rohit	Smith	1200	Bangalore	R240-1200	12-BGLO	**B217-00**
A4	Ravi	Smyth	1300	Bangalore	R470-1300	13-BGLO	**B217-00**

Sndx, DME are two phonetic functions and
Fi2D(C) ----- First two digits of C
La2D(C) ---- Last two digits of C

The important objective of indexing is to lessen the total number of comparisons by eliminating the records that are regarding non matches. Hence a technique is used for this approach and called blocking. In this technique, it divides the database into multiple blocks, so that the comparison happens only in a particular block instead of the whole database. A blocking key is used for this purpose. Based on their similarity, records are pushed into the blocks. Functions such as Double Metaphone, NYSIIS and Soundex are used to create blocking keys. In Table.1, it is explained in detail.

III. Indexing Techniques

The indexing techniques provide two stages in the process of data linkage. Build and Retrieve are the two stages in it.

Build: The records that present in each and every database are completely read, Blocking Keys are produced accordingly, and these records are pushed into particular indexes. Mostly a data structure called inverted index is selected for this purpose. The Blocking Keys then act as keys of inverted index and unique identifiers of the records that contain equal blocking keys are pushed into the same data structure. Fig. 2 shows an example for it. When associating two or more databases, a distinct data structure for every database can be constructed or only one data structure with same keys can be created.

Identifiers	**Initials**	**Blocking keys(Encoded)**
A1	Rahul	R490
A2	Smith	S520
A3	Jain	J170
A4	Raahul	R490
A5	Smyth	S520
A6	Rahull	R490
A7	Smith	S520
A8	Smyth	S520

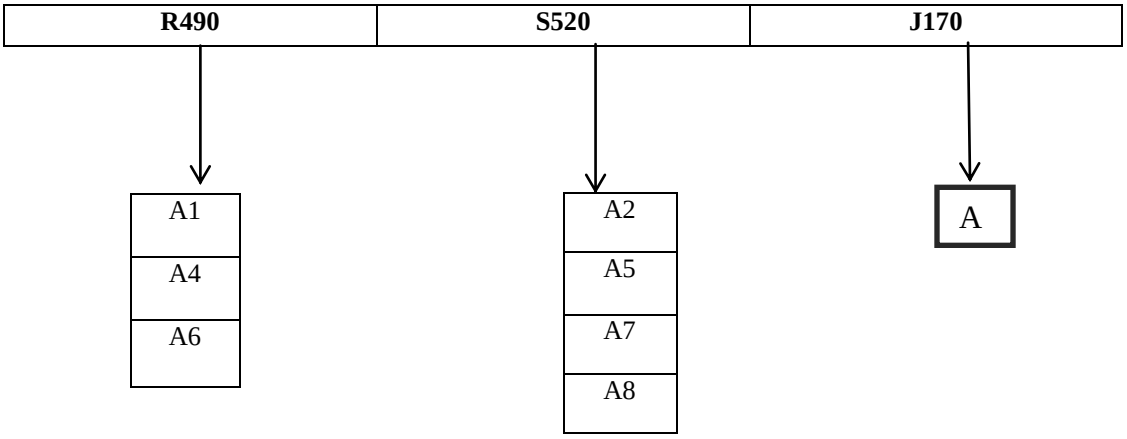

Fig. 2 Records with initials and encoded keys and equivalent inverted index data structure

Retrieve: For every block, a group of unique identifiers are retrieved from the data structure, and then candidate pairs are created using this. In the process of data linkage, the records present in the block of a single database are paired with the records from remaining databases that contain equal blocking key. If it is deduplication, records from the same block contain different pairs. As given in Fig. 2, R490 contain the pairs such as (A1, A4), (A4, A6), (A1, A6).

3.1. Traditional Blocking

It is employed competently by utilizing the inverted index data structure. The main disadvantage of this is flaws in the fields of records leads to incorrect blocking keys which in turn lead to wrong block placement of records. But this can be overwhelmed by consuming multiple descriptions for blocking keys. Another disadvantage of using this technique is block sizes created will be depending upon frequency delivery of blocking keys.

Candidate record pairs are generated by using;

$$R = k \times \binom{n_P}{k}\binom{n_Q}{k} = \frac{n_P n_Q}{k} \quad (1)$$

Where k is the number of distinct blocking keys,

n_P Is number of records in P

n_Q Is number of records in Q

$$R = k \times \frac{\binom{n_P}{k}\binom{n_P}{k-1}}{2} = \binom{n_P}{2}\binom{n_P}{k-1} \quad (2)$$

3.2 Q-Gram-Based Indexing

In this technique, the database is indexed so that records that contain related but not as it is as blocking keys are pushed into the same block. Hence the main goal of this is to generate variations for every blocking key using q-grams (subset of size q) and can push multiple unique identifiers into multiple blocks. Every blocking key is transformed into a group of q-grams and then sub lists are created up to a particular length that can be determined by the user. Hence these are transformed to strings and can be used as actual values in the data structure that is described in Fig. 3.

Identifiers	Blocking Keys	Sub-lists	Index values
A1	Rahul	[ra, ah, hu, ul], [ra, ah, hu], [ra, ah, ul], [ra, hu, ul], [ah, hu, ul]	**raahhuul**, raahhu, raahul, rahuul, ahhuul
A2	Raahul	[ra, aa, ah, hu, ul], [ra, aa, ah, hu], [ra, aa, ah, ul], [ra, aa, hu, ul], [ra, ah, hu, ul], [aa, ah, hu, ul]	raaaahhuul, raaaahhu, raaaahul, raaahuul, **raahhuul**, aaahhuul
A3	Rahull	[ra, ah, hu, ul, ll], [ah, hu, ul, ll], [ra, hu, ul, ll], [ra, ah, ul, ll], [ra, ah, hu, ll], [ra, ah, hu, ul]	Raahhuulll, ahhuulll, rahuulll, raahulll, raahhull, **raahhuul**

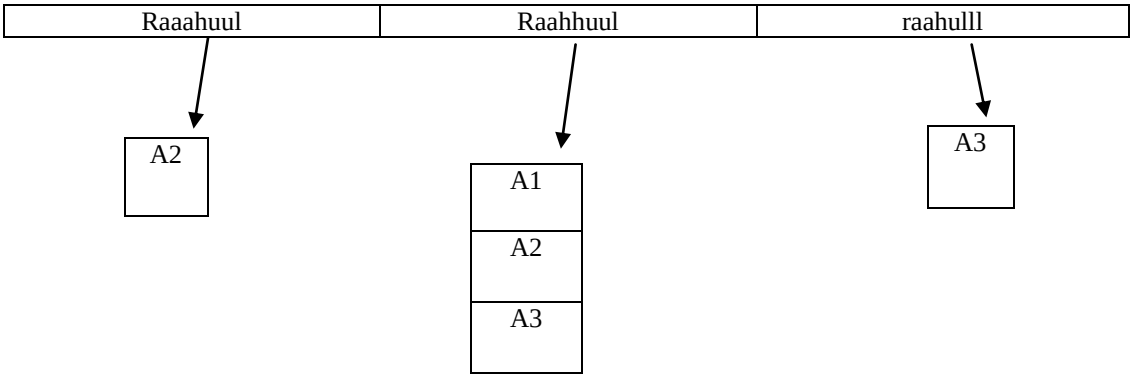

Fig.3. Q-Gram Technique considering bigrams (q=2) and representation using inverted index

3.3 Sorted Neighborhood Indexing

In this technique, the databases are sorted regarding the values of blocking keys, and then sequentially push a window of records above the sorted values. Then using that window, record pairs are generated. In two ways this technique is developed.

3.3.1 Sorted Array-Based Approach

The blocking keys are sorted according to their names and pushed into the array. Now move the window above this array, then record pairs are created. If the size of the array is $n_P + n_Q$, then the number of positions of window is $(n_P + n_Q - w + 1)$. But in the duplication it is $(n_Q - w + 1)$. This is represented in Fig.4.

Window positions	Blocking Keys	Identifiers
1	Rahul	A1
2	Raahul	A4
3	Rahull	A6
4	Jain	A3
5	Smith	A2
6	Smyth	A5
7	Smyth	A8
8	Smith	A7

Window Range	Record Pairs
1-3	(A1, A4), (A4, A6), (A1, A6)
2-4	(A4, A6), (A6, A3), (A4, A3)
3-5	(A6, A3), (A3, A2), (A6, A2)
4-6	(A3, A2), (A2, A5), (A3, A5)
5-7	(A2, A5), (A5, A8), (A2, A8)
6-8	(A5, A8), (A8, A7), (A5, A7)

Fig.4. Sorted Neighborhood Technique with a window size of 3

The disadvantage of this approach is if the chosen window length is very small, then all the records will not be covered with the unique blocking key. The best solution for this is combining more fields such that there can be a number of values. One more problem with this is sorting the blocking keys is perceptive to flaws and deviations in the values. The solution for this is to create multiple blocking keys or creating those keys by reversing the values. The time complexity of this is O (nlogn) where n=$n_P + n_Q$ for data linkage and n=n_P in case of deduplication.

3.3.2 Inverted Index-Based Approach:

Instead of pushing into sorted array, a data structure called inverted index is used in this approach. This is same as above approach but one difference between both the approaches is the comparison happens only once. There are some drawbacks in this technique. One of the drawbacks is number of blocks are dominating the record pairs created. The other drawback is blocking keys are sorted assuming that beginning is flawless. This technique is described by using the following figure Fig. 5. The one with same blocking keys represent different identifiers instead of in above approach.

Window positions	Blocking Keys	Identifiers
1	Rahul	A1
2	Raahul	A4
3	Smith	A2, A7
4	Jain	A3
5	Rahull	A6
6	Smyth	A5, A8

Window Range	Record Pairs
1-3	(A1, A4), (A1, A2), (A1, A7), (A4, A2), (A4, A7), (A2, A7)
2-4	(A4, A2), (A4, A7), (A2, A7), (A4, A3), (A2, A3), (A7, A3)
3-5	(A2, A7), (A2, A3), (A7, A3), (A2, A6), (A7, A6), (A3, A6)
4-6	(A3, A6), (A3, A5), (A3, A8), (A6, A5), (A6, A8), (A5, A8)

Fig.5. Inverted Index Approach

3.4 Indexing Based On Suffix-Arrays

In this indexing technique, push the blocking keys even suffixes into suffix-based data structure. The group of characters or strings in sorted order is nothing but Suffix Array. If the length of the string is 'k', then the number of suffixes is (k-l+1) where l is minimum length. The disadvantage of this technique is flaws and deviations at the end of blocking keys lead to records that are pushed into multiple blocks instead of the same block. The solution for this is to create not only the real suffixes for blocking keys, but also to create substrings to a particular length. This can be explained in detail with the help of Fig. 6. The block size is nothing but a number of identifiers.

Identifiers	**Blocking Keys**	**Suffixes**
A1	Fluorine	fluorine, luorine, uorine, orine, rine, ine
A2	Chlorine	chlorine, hlorine, lorine, orine, rine, ine
A3	Iodine	iodine, odine, dine, ine
A4	Amoxine	amoxine, moxine, oxine, xine, ine

Suffix	**Identifiers**
amoxine	A4
chlorine	A1,A2
dine	A3
fluorine	A1
hlorine	A2
~~ine~~	~~A1,A2,A3,A4~~
lorine	A2
Luorine	A1
Moxine	A4
Odine	A3
Orine	A1,A2
Oxine	A4
Rine	A1,A2
Uorine	A1
Xine	A4

Fig.6. Suffix-Based Approach with length l=3 and maximum block size=3. The block containing suffix ine is removed as it contains more block identifiers.

3.5 Canopy Clustering

Clusters are generated in such a way that the similarities between blocking keys are calculated using cosine or Jaccard measure. These are dependent upon tokens that can be words or characters. It can be employed efficiently using a data structure called inverted index. By translating blocking keys into a group of tokens, the data structure is developed whereas the key is the particular token. The records that contain the particular token in blocking key are added to the data structure. Two frequencies such as term and document are calculated in this approach. Term frequency is nothing but the presence of the token in the number of records. Document frequency is how often the token appears is calculated. In Fig.7, this approach is represented.

Identifiers	Blocking keys	Sorted Lists
A1	Ranjan	[(an, 2), (ja, 1), (nj, 1), (ra,1)]
A2	Raghanan	[(ag, 1), (an, 2), (gh, 1), (ha, 1), (na, 1), (ra, 1)]
A3	Pragranth	[(ag, 1), (an, 1), (gr, 1), (nt, 1), (pr, 1), (ra, 2), (th, 1)]

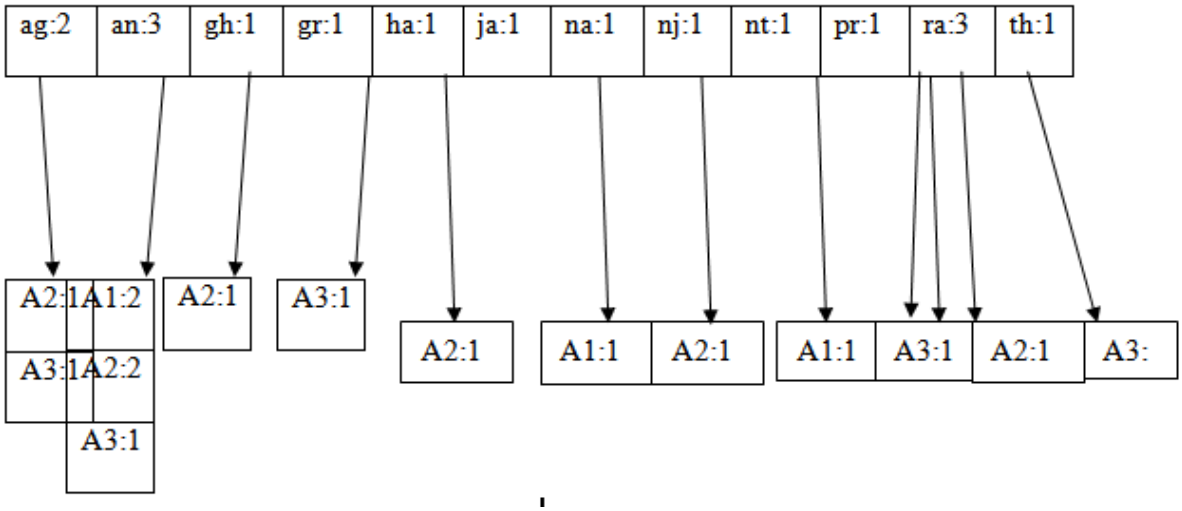

Fig.7.Canopy Clustering and Bigram list containing DF counts with inverted index data structure

IV. Experimental Results

There are some measures to evaluate the complexity of indexing and even quality. Let M_n and N_n be the number of equal and unequal record pairs. T_M and T_N are the true equal and unequal record pairs such that $T_M + T_N \leq M_n + N_n$.

By using one of the indexing techniques, some of the pairs of records are eliminated by applying a measure called Reduction Ratio represented by RR. It can be represented as

$$RR = 1.0 - \frac{T_M + T_N}{M_n + N_n} \quad (3)$$

The ratio of number of true equal record pairs to total number of equal pairs is called Pairs Completeness represented by PC as follows:

$$PC = \frac{T_M}{M_n} \quad (4)$$

The ratio of number of true equal record pairs to total number of record pairs is called Pairs Quality, simply represented as PQ which is as follows.

$$PQ = \frac{T_M}{T_M + T_N} \quad (5)$$

If PQ value is high then the indexing technique is best in terms of efficiency and creates a number of truly matched records and in the same way if PQ value is low then more non-matches are created.

The f-score is used to calculate the harmonic mean of Pairs Completeness and Pairs Quality and it can be represented as follows:

$$f = \frac{PC \times PQ}{PC + PQ} \quad (6)$$

Hence by using these measures, performance is evaluated.

From Table 2, it is clearly observed that q-gram based indexing technique is very slow when compared to other techniques. Array based sorted neighborhood and even traditional blocking are very fast by comparing with others.

Table 2. Runtimes Evaluation for every indexing technique per candidate record pair

Indexing Technique	Time in ms per candidate record pair	
	Minimum	Maximum
Traditional Blocking	0.002	0.972
Q-gram Based Indexing	0.005	163,484.394
Array Based sorted neighborhood Inverted Index based sorted neighborhood	0.011 0.002	0.288 3.040
Indexing Based On suffix-Arrays	0.024	168.561
Canopy Clustering	0.003	380.214

V. Conclusion

In this paper, five indexing techniques are discussed. The candidate record pair generation is discussed for every technique, based on that complexity is analyzed. Only by defining the blocking keys perfectly, accurate indexing and efficiency is obtained. Always the data are divided into non matches and matches so that it is easy to identify the blocking keys accordingly. Future work in this is to develop more new efficient techniques so that comparisons between records in a particular block should have very less similarity.

References

[1] A. Aizawa and K. Oyama, "A Fast Linkage Detection Scheme for Multi-Source Information Integration,"Proc. Int'l

Workshop Challenges in Web Information Retrieval and Integration (WIRI '05), 2005.

[2] A.K. Elmagarmid, P.G. Ipeirotis, and V.S. Verykios, "Duplicate Record Detection: A Survey,"IEEE Trans. Knowledge and Data Eng., vol. 19, no. 1, pp. 1-16, Jan. 2007.

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

[4] E. Rahm and H.H. Do, "Data Cleaning: Problems and Current Approaches,"IEEE Technical Committee Data Eng. Bull., vol. 23, no. 4, pp. 3-13, Dec. 2000.

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

[6] W.W. Cohen, P. Ravikumar, and S. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks,"Proc. Workshop Information Integration on the Web (IJCAI '03), 2003