Automated Selection of Blocking Columns for Record Linkage

K. Hima Prasad, Snigdha Chaturvedi, Tanveer A. Faruquie, L. Venkata Subramaniam, Mukesh K. Mohania
IBM Research India, New Delhi, India
{hkaranam,sncatur,ftanveer,lvsubram,mkmukesh@in.ibm.com}

Abstract—

Record Linkage is an essential but expensive step in enterprise data management. In most deployments, blocking techniques are employed which can reduce the number of record pair comparisons and hence, the computational complexity of the task. Blocking algorithms require a careful selection of column(s) to be used for blocking. Selection of appropriate blocking column is critical to the accuracy and speed-up offered by the blocking technique and requires intervention by data quality practitioners who can exploit prior domain knowledge to analyse a small sample of the huge database and decide the blocking column(s). However, the selection of optimal blocking column(s) can depend heavily on the quality of data and requires extensive analysis. An experienced data quality practitioner is required for the selection of optimal blocking columns. In this paper, we present a data-driven approach to automatically choose blocking column(s), motivated from the modus operandi of data quality practitioners. Our approach produces a ranked list of columns by evaluating them for appropriateness for blocking on the basis of factors including data quality and distribution. We evaluate our choice of blocking columns through experiments on real world and synthetic datasets. We extend our approach to be employed in scenarios where more than one column can be used for blocking.

I. Introduction

Real world data is often collected from eclectic sources. Each source can accumulate data in an independent fashion, and without following a consistent standard. Also, in most scenarios, the database lacks any attribute which can be treated as a unique global identifier. Due to this inherent nature of data collections, duplicate records, representing the same entity within the same database or across databases, can get introduced in the combined database. Identification of duplicates is essential for providing a standardized and consumable master view of the database. For example, an enterprise may have several departments for offering different services to its customers. Each department might maintain a separate database of customer details. It is possible for a customer to avail more than one service from the enterprise and hence, his details would appear in individual databases of the corresponding enterprise departments. However, because of various errors, like typographical mistakes, missing values, etc., that get introduced in the data collection process and lack of unique identifiers, it can be difficult for the enterprise to link the details of the customer across various databases of departments. Hence, Record Linkage, also known as De-duplication, becomes a critical step in data cleansing procedures[7], [5] and single view creation of entities.

In the absence of any prior knowledge, record linkage involves comparing each record in the database(s) with every other record using a comparison function and classifying them as duplicates or non-duplicates or possible duplicates based on the results of the comparison. This suggests two major challenges associated with record linkage: the linkage accuracy and the time complexity of the task.

Linkage accuracy primarily depends on the choice of comparison function and the classification model (that assigns pairs as duplicates/non-duplicates/possible duplicates) used for decision making. The computer science research community has made significant inroads in this direction, which now equip us with sophisticated comparison methods and elaborate learning models that can be applied for classification.

The brute force method of linkage includes comparing every record with millions of other records in the database(s). This results in the number of record pair comparisons growing quadratically with the total number of records. If there are two databases A and B across which record linkage has to be performed then the number of record pair comparisons turns out to be $O(|A| \times |B|)$. Similarly in case of record linkage within a database, A, the number of record pair comparisons would be $O(|A|^2)$. In many real life deployments it is impossible to allocate this much time and processing power for record linkage. Hence in all practical applications, regular record linkage is preceded by Blocking.

During blocking, the dataset is split into mutually exclusive 'blocks'. Records that are candidate matches are clubbed together into same block and so while searching for the duplicates of a record, it will be sufficient to compare it with other records within the same block. Blocking is a critical step in record linkage mechanism because the choice of the blocking scheme directly affects the performance of the record linkage procedure. If due to an incorrect choice of blocking scheme, two duplicates get assigned to different blocks then they will never be compared (since the blocks are mutually exclusive, and record pair comparisons are done within individual blocks).

Several blocking techniques have been proposed to identify candidate matches accurately and quickly with relatively lesser computation complexity. Most of these techniques rely on selection of an appropriate blocking column (or a set of columns) based on which the whole record is put in a block. Ideally, blocking columns should be highly discriminative columns whose values can be used to decide which records

978-1-4673-2401-4/12/$31.00 ©2012 IEEE

are candidate matches and hence should be placed in the same block. For example, *Standard Blocking* [10], is one such popular blocking method in which a blocking column is selected and a blocking key is defined on this blocking column using its values directly or after applying some transformation functions like Soundex [15] [16], NYSIIS [19], or Metaphone [14].

While much attention has been paid on designing quicker and more accurate blocking schemes, identification of blocking column(s) has attracted significantly little attention, and requires intensive human intervention. In practice, blocking column(s) is meant to be selected by trained data quality practitioners who examine a small sample of the complete database(s) and exploit their domain knowledge and experience in deciding the blocking column(s). In most practical situations, analysing the complete database is impossible for the practitioner and so they work with smaller samples from the database. Due to their limited size that is humanly examinable, smaller samples fail to accurately describe the properties of the complete database(s). This can result in decisions being taken based on prior experience instead of quality of data. Thus, the performance of blocking (and hence record linkage) becomes dependent on the wisdom of these data quality practitioners. The situation worsens in case of record linkage being performed by a common business end user who might lack prior domain knowledge and might be unaware of the data quality of the database(s).

In this paper, we present a data driven method to automatically select blocking column(s). The approach is closely aligned with the modus operandi of data quality practitioners. Our method analyses the various columns present in the database and ranks them by quantifying their usability as a blocking column using a measure specifically designed to address the above mentioned issues. The measure evaluates the columns on the following criteria:

1) Completeness of the column: Column(s) having lots of missing values fail to be sufficiently informative for blocking
2) Number of blocks which will be produced (b) when the column is used for blocking: Column(s) producing very few blocks don't offer any significant speed-ups
3) Uniformity of the distribution of records across the blocks produced: Highly skewed distributions result in speed-up being dominated by the heavily occupied columns
4) Mutual independence of columns in case of two or more blocking columns: Columns dependent on each other don't add any value to the blocking key as compared to one of them

We have validated our method on both real-world and artificial datasets.

II. Block Selection Approach

The main goal of our approach is to evaluate individual columns (or a combination of them called a 'column-set') for their suitability for blocking and rank them in decreasing order.

For the sake of simplicity, we explain our method for a single database. However, it is easy to see how our method can be extended to more than one database by considering all the records at once as a single consolidated database.

A. Single Column Blocking

In this section we present our method to rank columns based on their Blockability for the Single Column Blocking case.

In addresses, columns like *Zip Code* or *State Name* might seem very intuitive for blocking but might not be usable if their quality and distribution is not good. This suggests that selection of blocking column is not solely domain dependent but is data driven. Columns that work well for blocking on a particular database might not perform well on another database from the same domain. Hence, it is advisable to decide the blocking column after analysing the data instead of exploiting domain knowledge. Based on this idea, we adopt the following summarized approach to solve this problem:

- For each column, C, of the database, D:
  - Compute *Blockability*, $B(C)$
- Rank all columns of D in decreasing order of $B(C)$. The top ranked column can be used for blocking.

1) Block and Block Size: Given a column, C, each of the unique values of the column can form an independent block. Hence, the number of blocks formed by a given column, C, would essentially be equal to the number of unique values in C. Also, the corresponding block size would be the fraction of records that belong to that block when binning is done based on column C. Block size of a block, i, is defined as:

$$\text{block size}(i) = \frac{N_i}{N}$$

where, N_i = Number of records that belong to block i
N = Total number of records in the database.

2) Factors determining Blockability: Blockability of a column is a measure designed in this work to quantify its usability for blocking. The appropriateness of a particular column, C, for blocking depends on the following factors:

1) Completeness of C
2) Distribution of records across the blocks formed by C
3) Number of blocks formed by C

Completeness of C: This is the most important criterion that needs to be taken care of as it directly affects the accuracy of blocking and hence, the record linkage algorithm. A blocking column should have as few missing values as possible. Since record pair comparisons are done within individual blocks only, records with missing values never get compared to their potential duplicates. We define the following score, $\alpha(C)$ to measure completeness of a column, C:

$$\alpha(C) = e^f$$

where f = fraction of records having a missing value in C

For a column to be good for blocking, we want its $\alpha(C)$ to be low.

Distribution of records across the blocks formed by C: This factor also plays a very important role in determining

the speed up offered by the blocking method. For example in case of Standard Blocking, for obtaining the ideal speed-up offered by a blocking column it is assumed that each block will have n/b records. This is, in practice, hardly achievable and depends on the block having greatest number of records. So, we want the distribution of records across blocks to be as uniform in nature as possible. We propose to use entropy [17] of block sizes to ensure uniformity of distribution. The key idea is that for a distribution to be good for blocking, all the blocks should have approximately same block size i.e. the entropy of block sizes should be high. For a given C:

$$\beta(C) = -\sum(p_{\text{block size(i)}} * ln(p_{\text{block size(i)}}))$$

where,

$p_{\text{block size(i)}} = \frac{\text{block size(i)}}{\sum_j^{\bullet} \text{block size(j)}}$

block size(i) = size of i^{th} block

b = Number of blocks produced on using C for blocking

Number of blocks formed by C: The speed up obtained by blocking is due to the fact that a big dataset is divided into several subsets (or blocks) and each block is worked upon independently. This is what reduces the complexity of the task. Understandably, if the number of blocks if too small then the purpose of blocking gets defeated and there is no significant improvement in speed or complexity.

We define $\gamma(C)$, in the form of a sigmoid function, to capture this information. $\gamma(C)$ is designed such that its value will be very small if the number of blocks formed is small and so columns leading to fewer bins will be penalized. $\gamma(C)$ is defined as:

$$\gamma(C) = \frac{1}{1+e^{-(b-5)}}$$

where, b is the number of blocks formed by column, C.

For a column to be a good for blocking, the value of $\gamma(C)$ should be high.

3) Blockability: Having defined measures to quantify the factors on which Blockability of a column, C, depends, we now combine these measures to define Blockability, $B(C)$. It would not be advisable to use these measures directly as each one of them would have different ranges of values and so $B(C)$ would get dominated by the higher ranged measure. So, we normalize each of these three measures as follows:

$$Norm(M(C)) = \frac{M(C)}{\sum_{\text{all columns}} M(C)}$$

where $M(C)$ is either $\alpha(C)$ or $\beta(C)$ or $\gamma(C)$.

Now, we define $B(C)$ using these normalized measures. As explained earlier, for a good blocking column $\alpha(C)$ should be low and $\beta(C)$ and $\gamma(C)$ should be high.

So $B(C)$ of a column, C, is defined as:

$$B(C) = \frac{Norm(\beta(C)) + Norm(\gamma(C))}{Norm(\alpha(C))}$$

For a column to be good for blocking, value of $B(C)$ should be high.

4) Ranking: Initially, all columns are considered as potential blocking column and so Blockability values, $B(C)$, are computed for each column, C. Thereafter, the columns are sorted in decreasing order of corresponding $B(C)$ values to produce a ranked list of blocking columns.

B. Multi-column Blocking

In Multi-column Blocking, a set of columns is used for blocking instead of a single column. We refer to this set of columns as a 'column-set' and its members as 'member-columns'. While blocking, all the member-columns are considered and the blocking key is defined by combining their values using operator such as AND. Like Single Column Blocking, domain knowledge and prior experience based selection of blocking column (a column-set in this case) does not necessarily work. Hence, there is a need to do the selection based on data analysis. We adopt the following summarized approach to automatically select blocking columns:

- Choose a maximum column-set size, say l
- Form column-sets using various columns of the database taking i columns at a time where i ranges from 2 to l
- For each column-set, C, of the database, D:
  - Compute *Blockability*, $B(C)$
- Rank all column-sets of D in decreasing order of $B(C)$

1) Block and Block Size: In case of Multi-column Blocking, usually two or three columns are used for blocking. So, the value of l is set to be 2 or 3.

For given a column-set, C, each of the unique combinations of member-column values joined with AND operator form an independent block.

Blockability of a column-set, C, depends on the following factors:

1) Completeness of C
2) Distribution of records across the blocks formed by C
3) Number of blocks formed by C
4) Mutual Independence of the member-columns of C

The first three factors are same as in case of Single Column Blocking.

Mutual Independence of columns: This factor is relevant in case of Multi-column Blocking only. When a blocking key is defined using a column-set, each member-column has its own share of contribution. If one of the members is highly dependent on (or is derived from) another member then adding the second member to the column-set would not add any more information even though, individually, both of them may serve as excellent blocking columns. We propose to use Mutual Information (MI) [4] to quantify this factor. Mutual Information can be used to measure the mutual dependence between two random variables. Here we use it to measure the degree of dependence between the member-columns of a column-set. We define $\delta(C)$ for a column-set, C, as the measure of dependence between its members as:

$$\delta(C) = \text{MI between member-columns}$$

If the member-columns are totally independent then the value of MI will be 0 and hence $\delta(C)$ will be 0. If member-columns are completely dependent then MI will be equal to the entropy

Dataset description	No. of records	No. of duplicates clusters	True duplicates	Column Names
Mailing List (Synthetic)	10,000	500	known	SSN, First Name, Middle Initial, Last Name, Street Number, Street Name, Apartment Number, City, State, Zip Code
Indian Postal Addresses (Real World)	5,000	not known	not known	Door No., Floor No. Building Name, Building Type, Unit Name, Street Name, Street Type, Landmark Name, Landmark Type, Area Name, District, State Name, Zip Code

TABLE I
DATASETS DESCRIPTION

of either member. For C to be appropriate for blocking, value of $\delta(C)$ should be low.

2) Blockability: As described in Section II-A3 we first normalize all the four measures namely: $\alpha(C)$, $\beta(C)$, $\gamma(C)$ and $\delta(C)$.

In case of Multi-column Blocking, $B(C)$ needs to incorporate $\delta(C)$. So for Multi-column Blocking, it is defined as:

$$B(C) = \frac{Norm(\beta(C))+Norm(\gamma(C))}{Norm(\alpha(C))+Norm(\delta(C))}$$

Understandably, a higher value of $B(C)$ suggests greater suitability of C for blocking.

3) Ranking: After computing the Blockability, all the column-sets are sorted in decreasing order of corresponding $B(C)$ values.

III. EMPIRICAL EVALUATION

A. Automatic Selection of Blocking Column(s)

1) Datasets: For our experiments we have used two separate datasets.

The first dataset is synthetic, and contains mailing addresses generated using UIS DBGen [8], [9]. This dataset has information on true duplicates (i.e. information that which record pairs are duplicates). Details of the dataset are available in Table I. No additional standardization or cleaning was done on this dataset.

The second dataset consists of postal addresses of customers from a telecom service provider. The addresses were already standardized and stored in the form of a relational database (details in Table I). Since this is real world data, obtaining the true duplicates/matches information for it was difficult.

2) Evaluation Metrics: Since blocking affects the speed and accuracy of the record linkage algorithm, Reduction Ratio (RR), Pair Completeness (PC) and F measure have been used in past to measure blocking performance [1]. We also use these measures to evaluate the performance of blocking techniques and hence, the appropriateness of blocking column(s). Calculation of PC and F-score require knowledge of true duplicates beforehand while calculation of RR can be done without this information.

Reduction ratio (RR) [3] is defined as:

$$RR = 1 - \frac{N_b}{n^2}$$

where,
N_b is the number of record pair produced by the blocking method for comparison,
n is the total number of records in the database and so n^2 is the total number of record pairs possible.
The reduction ratio measures the relative reduction of the comparison space.

Pair Completeness (PC) [3] is defined as:

$$PC = \frac{s_M}{N_M}$$

where, s_M is the number of correctly classified true matched record pairs in the blocked comparison space,
N_M is the the total number of true match record pairs in the complete dataset.

F-score [3] combines RR and PC and is defined as their harmonic mean:

$$F - score = 2(\frac{PC \times RR}{PC + RR})$$

Apart from the above mentioned measures used to measure blocking performance, we use **Kendall's tau coefficient** [12] to compare the ranking of columns produced using our method with gold rankings (described later in this section). If D is the set of items that are to be ranked, a (strict) ranking r is a binary relation over $D \times D$ such that for every pair of items $(d_i, d_j) \in D \times D$, either $(d_i, d_j) \in r$ or $(d_j, d_i) \in r$. Kendall's coefficient for two ranking r_a and r_b is defined as:

$$\tau(r_a, r_b) = \frac{\text{No. of concordant pairs - No. of discordant pairs}}{\text{No. of concordant pairs + No. of discordant pairs}}$$

A pair, (d_i, d_j), is said to be concordant iff $(d_i, d_j) \in r_a$ and $(d_i, d_j) \in r_b$ and discordant otherwise.

The value of $\tau(r_a, r_b)$ will be 1 if the two rankings are same, -1 if one is reverse of other and 0 if they are independent.

3) Comparison Method: To validate our blocking column(s) selection method, we use our method to produce a list of all columns (or column-sets) ranked in decreasing order of their suitability for use in blocking (Blockability). Our aim is now to evaluate the goodness of this ranking by comparing it to a ranking depicting true blocking performance.

For this, we perform record linkage on the same dataset using a different column (or column-set) for blocking each time. For example, if there are K columns in the dataset then say, l, columns (l=1 for Single Column Blocking and $l > 1$ for Multi-column Blocking) are used for blocking and the rest $K - l$ are compared using some comparison measure followed by classification based on comparison results. For each blocking column (or column-set) we measure the corresponding RR, PC and F-score of Record Linkage. Subsequently, we generate three different ranked lists obtained by ranking all the columns (or column-sets) according to the three measures defined above. This can be summarized as:

- For every column (or column-set), C, in the database:
  - Select C for blocking
  - Apply the comparison measure on the remaining columns
  - Use a classification algorithm to determine duplicates
  - Observe RR, PC and F-score values
- For each of the three measures (RR, PC and F-score):
  - Rank all the columns (or column-sets) in order of the measure (RR, PC or F-score)

Correlation Coefficient	Measure used for gold ranking		
	RR	PC	F-score
Synthetic Dataset			
Kendall's tau	0.822	0.778	0.733
Pearson's	0.915	0.879	0.854
Real World Dataset			
Kendall's tau	0.778	–	–
Pearson's	0.923	–	–

TABLE II
CORRELATION VALUES FOR SINGLE COLUMN BLOCKING

Rank	Ranking by our method	Measure used for gold ranking		
		RR	PC	F-score
1	Street Name	Street Name	Street Name	Street Name
2	Last Name	Last Name	Last Name	Last Name
3	First Name	First Name	First Name	First Name

TABLE III
TOP COLUMNS FOR SINGLE COLUMN BLOCKING ON SYNTHETIC DATA

Since our aim is to evaluate only the blocking capability of the columns under consideration, we ensure that the comparison and classification methods for each iteration remains identical (details mentioned later in this section).

Finally, the three rankings obtained above can be treated as separate gold rankings each measuring a different aspect of blocking column's performance: RR, PC and F-score. We compare the ranking produced by our method with each of the three gold rankings using Kendall's tau correlation coefficient. Apart from Kendall's tau coefficient, we also use Pearson's correlation coefficient [18]. The general idea is that if our method selects good blocking column(s) then the ranking produced by it will be similar to the gold rankings and hence the correlation values will be close to 1.

4) Record Linkage Software: There exist several record linkage systems like *Febrl* [2], *AutoMatch* [11] and *TAILOR* [6]. For our experiments, we have used *Febrl* which is a freely available tool having several standard algorithms for performing data standardization and record linkage including blocking. We have used *Febrl* to perform record linkage on standardized (relational) databases using Standard Blocking (without using any transformation function) as the blocking method and Edit distance [13] as the comparison measure. For classification, we have used Optimal Threshold when true duplicates information was known (synthetic data) and K-means when this information was not available (real-world data).

5) Results for Single Column Blocking using Synthetic Dataset: Table II lists the correlation values when the ranking produced by our method was compared to the gold ranking produced using RR, PC and F-score respectively. We see that in each comparison, the correlation value is close to 1 which suggests that our method judges the blocking capability of each columns correctly.

Furthermore, Table III lists the top three columns as determined by each of the four rankings. We see that the same three columns appear in all the rankings and their respective ranks produced by our method is exactly same as that with gold rankings.

6) Results for Single Column Blocking using Real World Dataset: We did not have the true duplicates information for the real world data containing postal addresses. Due to lack of this information, PC and F-score values could not be calculated while performing record linkage on this dataset (described in Section III-A3). So in this case, we have compared the ranking produced by our method with the gold ranking based on RR only. Table II lists the correlation values obtained for this dataset.

Since this real world data was obtained from an ongoing data quality project, we could consult data quality practitioners working on this data for human validation of our results. The experts agreed that the top ranked column (out of a total of 12 columns) produced by our method (*Zip Code*) was indeed used by them for blocking. Interestingly, they had arrived at this results after several 'hit-and-trial' operations of record linkage using some column and careful manual analysis of corresponding results.

Most record linkage softwares include an 'Explore' phase as a preprocessing step which provides several quickly computable statistical information about the data including number of unique values and number of missing values in each column. This is done to help the user decide a blocking column. However, intuition is often misleading, and there is a need for extensive data quality analysis of the column. Skipping this analysis can lead to poor blocking, or a need for numerous 'hit-and-trial' iterations as in this dataset. This dataset provides a good example of such a case. For example, looking at just the number of unique values and missing values information, *State Name* seems to be a better candidate for blocking than *Zip Code* as it had fewer unique values (*State Name* has 22 and *Zip Code* has 686) as well as lesser missing values (*State Name* has 81 while *Zip Code* has 235 missing values) . However, our method ranked *Zip Code* higher than *State Name*. This happened because the $\beta(C)$ value for *State Name* (0.567) is much lower than that of *Zip Code* (0.874). Figure 1 further illustrates that *State Name* with nearly 60% of the records belonging to one block is not a good blocking column.

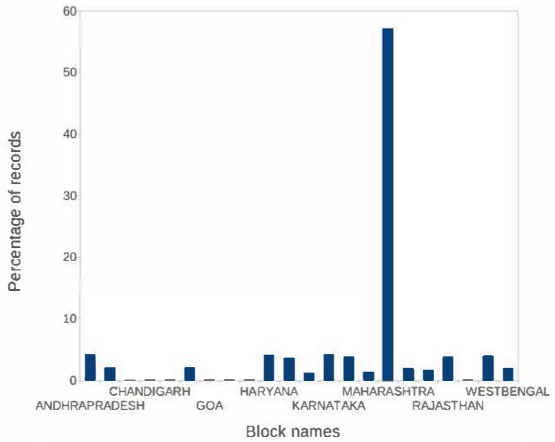

Fig. 1. Distribution of records across blocks for *State Name* for Real World Data

7) Results for Multi-column Blocking: We have used the synthetic data to validate our approach proposed for multi-

	Measure used for gold ranking		
Correlation Coefficient	**RR**	**PC**	**F-score**
Synthetic Dataset			
Kendall's tau	0.801	0.715	0.715
Pearson's	0.949	0.895	0.895

TABLE IV
CORRELATION VALUES FOR MULTI-COLUMN BLOCKING

Rank	Ranking by our method	Measure used for gold ranking		
		RR	PC	F-score
1	Last Name AND Street Name	Last Name AND Street Name	First Name AND Street Name	First Name AND Street Name
2	First Name AND Street Name	First Name AND Street Name	Last Name AND Street Name	Last Name AND Street Name
3	First Name AND Last Name	First Name AND Last Name	Street Name AND Apartment Number	Street Name AND Apartment Number

TABLE V
TOP COLUMNS FOR MULTI-COLUMN BLOCKING ON SYNTHETIC DATA

column blocking. For our experiments, we have used a maximum column-set size of 2. Since there are 10 columns in this dataset, the number of column-sets formed was 45 ($\binom{10}{2}$). Table IV compares the correlation values obtained when the ranking produced by our approach was compared to the ranking obtained by sorting according to RR, PC and F-score. We observe that, as for Single Column Blocking, the correlation values are fairly close to 1 indicating that the proposed method is a fair technique to automatically select blocking column-sets also.

Additionally, Table V compares three top ranked column-sets from each of the 4 rankings. We observe that the first two columns proposed by our method appear in top three of each of the three gold rankings. Also, one of the gold rankings (ranking obtained using RR) contains all the three columns in same order as proposed by our approach.

IV. CONCLUSIONS

Selecting blocking column(s) is a non-trivial task for business enterprises, requiring intensive human judgement. Typically, the choice of the blocking column(s) depends on a small subset of the databsase, which data quality practitioners examine to estimate the data quality. Often, the examinable sample size is so small (typically few hundreds) that it fails to be sufficiently representative of the entire database containing millions of records. The diverse factors to be taken care of during analysis also make the procedure more complicated. Typically, decisions are made based on prior experience and a 'hit-and-trial' approach is followed where seemingly good columns are iteratively chosen for blocking; and the results manually analysed on basis of measures like reduction ratio and a rough judgement of duplicates quality. This requires domain expert involvement and a lot of time and effort. Also, selection of blocking column(s) is so heavily dependent on data quality that decisions based on simple statistical facts sometimes don't work well.

The current paper is an attempt towards automating this complex process, which can bring down the cost of record linkage systems considerably. We find this to be an interesting research problem which, to the best of our knowledge, has not attracted sufficient attention. One of the contributions of this paper is to address this problem and bring it to the notice of the research community. In this paper, we have presented a method to automatically select blocking column(s). Our information-theoretic data driven approach ranks the columns of a given database in order of their suitability for blocking. We propose a measure to quantify the 'Blockability' which is intuitive and takes into account missing values, number of blocks formed, distribution of record across them and correlation between columns. Our results for selection of blocking column(s) involving both single and multiple columns show high correlations with gold rankings.

REFERENCES

[1] R. Baxter and P. Christen. A comparison of fast blocking methods for record linkage. In *In ACM SIGKDD workshop on Data Cleansing, Record Linkage and Object Consolidation*, pages 25–27, Washington DC, 2003.

[2] P. Christen. Febrl -: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 1065–1068, New York, NY, USA, 2008. ACM.

[3] P. Christen and K. Goiser. Quality and complexity measures for data linkage and deduplication. In F. Guillet and H. J. Hamilton, editors, *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 127–151. Springer, 2007.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory, 2rd Edition*. Wiley Series in Telecommunications and Signal Processing, July 2006.

[5] M. N. Dani, T. A. Faruquie, R. Garg, G. Kothari, M. K. Mohania, K. H. Prasad, L. V. Subramaniam, and V. N. Swamy. A knowledge acquisition method for improving data quality in services engagements. In *IEEE SCC*, pages 346–353. IEEE Computer Society, 2010.

[6] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid. Tailor: A record linkage toolbox. In *International Conference on Data Engineering*, ICDE '02, 2002.

[7] T. A. Faruquie, K. H. Prasad, L. V. Subramaniam, M. K. Mohania, G. Venkatachaliah, G. Kulkarni, and S. Basu. Data cleansing as a transient service. In *Proc. of ICDE*, 2010.

[8] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 24:127–138, May 1995.

[9] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2:9–37, January 1998.

[10] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, June 1989.

[11] M. A. Jaro. Software demonstrations. In *In Proc. of an International Workshop and Exposition - Record Linkage Techniques*, Arlington, VA, USA, 1997.

[12] M. G. Kendall and J. D. Gibbons. *Rank Correlation methods*. 1962.

[13] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[14] L. Philips. Hanging on the metaphone. *Computer Language Magazine*, 7(12):39–44, December 1990.

[15] R. Russell, April 1918.

[16] R. Russell, November 1922.

[17] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[18] S. M. Stigler. Francis galton's account of the invention of correlation. *Statistical Science*, 4:73–79, 1989.

[19] R. L. Taft. Name search techniques. *New York State Identification and Intelligence System*, 1970.