Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 5, May 2014, pg.404 – 414

RESEARCH ARTICLE

Appraisal of Efficient Techniques for Online Record Linkage and Deduplication using Q-Gram Based Indexing

Dr. M.V Shiva Prasad, Ch.Krishna Prasad, B.Rambabu

Abstract: - We present new indexing techniques for approximate string matching. The index collects text q-samples, that is, disjoints text substrings of length q, at fixed intervals and stores their positions. At search time, part of the text is filtered out by noticing that any occurrence of the pattern must be reflected in the presence of some text q-samples that match approximately inside the pattern. The aim of this technique is to index the database such that records that have a similar, not just the same BKV (Blocking key value) will be inserted into the same block. Assuming the BKVs are strings, the basic idea is to create variations for each BKV using q-grams (sub-strings of lengths q), and to insert record identifiers into more than one block.

Index terms: - q-samples, substrings, BKV, q-grams, Record identifiers

1. INTRODUCTION

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. Linked data can also help in health surveillance systems to enrich data that is used for the detection of suspicious patterns, such as outbreaks of contagious diseases.

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, for example for collaborative marketing or e-Commerce projects. Many government organizations are now increasingly employing record linkage, for example within and between taxation offices and departments of social security to identify people who register for assistance multiple times, or who work and Collect unemployment benefits.

Other domains where record linkage is of high interest are fraud and crime detection, as well as national security. Security agencies and crime investigators increasingly rely on the ability to quickly access files for a particular individual under investigation, or crosscheck records from disparate databases, which may help to prevent crimes and terror by early intervention. The problem of finding records that relate to the same entities not only applies to databases that contain information about people. Other types of entities that sometimes need to be matched include records about businesses, consumer products, publications and bibliographic citations, Web pages, Web search results, or genome sequences. In bioinformatics, for example, record linkage techniques can help find genome sequences in large data collections that are similar to a new, unknown sequence. In the field or information retrieval, it is important to remove duplicate documents (such as Web pages and bibliographic citations) in the results returned by search engines, in digital libraries or in automatic text indexing systems. Another application of growing interest is finding and comparing consumer products from different online stores. Because product descriptions are often slightly varying, matching them becomes challenging.

1.1. The Record Linkage Process

Most real-world data are dirty and contain noisy, incomplete and incorrectly formatted information, a crucial first step in any record linkage or deduplication project is data cleaning and standardization. It has been recognized that a lack of good quality data can be one of the biggest obstacles to successful record linkage. 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.

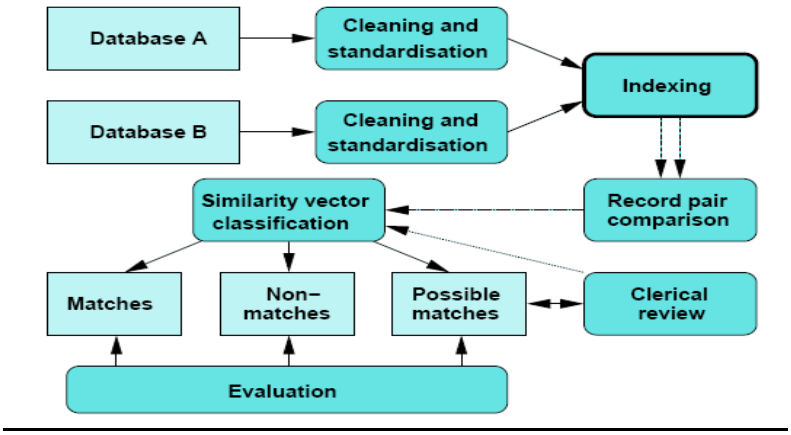


Fig.1. Outline of the general record linkage process

The indexing (Q-gram) generates pairs of candidate records that are compared in detail in the comparison step using a variety of comparison functions appropriate to the content of the record fields (attributes). Approximate string comparisons, which take (typographical) variations into account, are commonly used on fields that for example contain name and address details, while comparison functions specific for date, age, and numerical values are used for fields that contain such data. Several fields are normally compared for each record pair, resulting in a vector that contains the numerical similarity values calculated for that pair.

Using these similarity values, 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. Record pairs that were removed in the indexing step are classified as non-matches without being compared explicitly. The majority of recent research into record linkage has concentrated on improving the classification step, and various classification techniques have been developed. Many of them are based on machine learning approaches. 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. This is usually a time-consuming, cumbersome and error-prone process, especially when large databases are being linked or deduplicated. Measuring and evaluating the quality and complexity of a record linkage project is a final step in the record linkage process.

1.2. Indexing For Record Linkage and 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 $|A| \times |B|$ comparisons between two records (with $|\cdot|$denoting the number of records in a database). Similarly, when deduplicating a single database A, the maximum number of possible comparisons is $|A| \times (|A|-1)/2$, 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 [9], [32], making the na¨ıve approach of comparing all pairs of records not feasible when the databases are large. For example, the matching of two databases with one million records each would result in 1012 (one trillion) possible record pair comparisons.

At the same time, assuming there are no duplicate records in the databases to be matched (i.e. one record in A can only be a true match to one record in B and vice versa), then the maximum possible number of true matches will correspond to min(|A|, |B|). Similarly, for a deduplication the number of unique entities (and thus true matches) 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 quadratic ally as databases are getting larger, the number of potential true matches only increases linearly in the
size of the databases.

Given this discussion, 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 nonmatches. The traditional record linkage approach has employed an indexing technique commonly called *blocking*, which splits the databases into no overlapping blocks, such that only records within each block are compared with each other. A blocking criterion, commonly called a *blocking key* (the term used in this paper), is either based on a single record field (attribute), or the concatenation of values from several fields.

Because real-world data are often dirty and contain variations and errors, an important criterion 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. For strings that contain personal names, for example, phonetic similarity can be obtained by using phonetic encoding functions such as Soundex, NYSIIS or Double-Metaphone. These functions, which are often language or domain specific, are applied when the blocking key values (BKVs) are generated.

2. RELATED WORK

Approximate string matching is a recurrent problem in many branches of computer science, with applications to text searching, computational biology, pattern recognition, signal processing, etc. The problem is: Given a long text T1...n of length n, and a (comparatively short) pattern P1...m of length m, both sequences over an alphabet _ of size σ, retrieve all the text substrings (or "occurrences") whose edit distance to the pattern is at most k. The edit distance between two strings A and B, ed(A,B), is defined as the minimum number of character insertions, deletions and substitutions needed to convert A into B or vice versa. We define the "error level" as $\alpha = k/m$. Note that the problem is meaningful for $0 \leq \alpha < 1$, as otherwise the pattern matches everywhere.

In the on-line version of the problem, the pattern can be preprocessed but the text cannot. The classical solution uses dynamic programming and is O(mn) time [Sel80]. It is based on filling a matrix C0...m,0...n, where Ci,j is the minimum edit distance between P1...i and a suffix of T1...j . Therefore all the text positions j such that $C_{m,j} \leq k$ are the endpoints of occurrences of P in T with at most k errors. The matrix is initialized at the borders with $C_{i,0} = i$ and $C_{0,j} = 0$,while its internal cells are filled using

$C_{i,j}$ = if Pi = Tj then $C_{i-1,j-1}$ else 1 + min($C_{i-1,j}$,$C_{i-1,j-1}$,$C_{i,j-1}$)

which extends the previous alignment when the new characters match, and otherwise selects the best choice among the three alternatives of insertion, deletion and substitution. Fig. 1 shows an example. In an on-line searching only the previous column $C_{*,j-1}$ is needed to compute the new one $C_{*,j}$, so the space requirement is only O(m).

		s	u	r	g	e	r	y
	0	0	0	0	0	0	0	0
s	1	0	1	1	1	1	1	1
u	2	1	0	1	2	2	2	2
r	3	2	1	0	1	2	2	3
v	4	3	2	1	1	2	3	3
e	5	4	3	2	2	1	2	3
y	6	5	4	3	3	**2**	**2**	**2**

Figure 2.Dynamic programming matrix to search the pattern "survey" inside the text "surgery".

2.1. Contributions

While various indexing techniques for record linkage and deduplication have been developed in recent years, so far no thorough theoretical or experimental survey of such techniques has been published. Earlier surveys have compared four or less indexing techniques only. It is therefore currently not clear which indexing technique is suitable for what type of data and what kind of record linkage or deduplication application. The aim of this survey is to fill this gap, and provide both researchers and practitioners with information about the characteristics of a variety of indexing techniques, including their scalability to large data sets, and their performance for data with different characteristics.

The contributions of this paper are a detailed discussion of Q-gram indexing technique (with a total of four Variations of them), a theoretical analysis of their complexity, and an empirical evaluation of these techniques within a common framework on a variety of both real and synthetic data sets.

3. PROPOSED APPROACH

The aim of this technique is to index the databases such that records that have a similar, not just the same, BKV will be inserted into the same block. Assuming the BKVs are strings, the basic idea is to create variations for each BKV using q-grams (sub-strings of lengths q), and to insert record identifiers into more than one block.

3.1. Blocked Key Values

Each BKV is converted into a list of q-grams, and sub list combinations of these q-gram lists are then generated down to a certain minimum length, which is determined by a user-selected threshold t (t _ 1). For a BKV that contains k q-grams, all sub-list combinations down to a minimum length of l = max(1, bk × tc) will be created (b. . .c denotes rounding to the next lower integer number). These sub-lists are then converted back into strings and used as the actual key values into an inverted index, as is illustrated in Figure 3.

Different from the inverted index used in traditional blocking is that each record identifier is generally inserted into several index lists, according to the number of q-gram sub-lists generated for its BKV. With a threshold t = 1.0 however, each record identifier will be inserted into one inverted index list only, and in this case q-gram based indexing will generate the same candidate record pairs as traditional blocking.

Identifiers	BKVs (Surname)	Bigram sub-lists	Index key values
R1	Smith	[sm,mi,it,th], [mi,it,th], [sm,it,th], [sm,mi,th], [sm,mi,it]	**smmiitth**, miitth, smitth, smmith, smmiit
R2	Smithy	[sm,mi,it,th,hy], [mi,it,th,hy], [sm,it,th,hy], [sm,mi,th,hy], [sm,mi,it,hy], [sm,mi,it,th]	smmiitthhy, miitthhy, smitthhy, smmithhy, smmiithy, **smmiitth**
R3	Smithe	[sm,mi,it,th,he], [mi,it,th,he], [sm,it,th,he], [sm,mi,th,he], [sm,mi,it,he], [sm,mi,it,th]	smmiitthhe, miitthhe, smitthhe, smmithhe, smmiithe, **smmiitth**

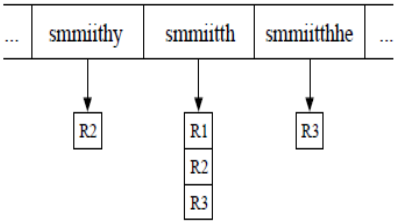


Figure 3. Q-gram based indexing with surnames used as BKVs, index key values based on bigrams (q = 2), and calculated using a threshold set to t = 0.8. The right-hand side shows three of the resulting inverted index lists (blocks), with the common BKV highlighted in bold in the index key value column.

Figure 3 illustrates q-gram based indexing for three example records, q = 2 (bigrams), and a threshold t = 0.8. The BKV 'Smith' in the first record (R1), for example, contains four (k = 4) bigrams: 'sm', 'mi', 'it', 'th' (assuming all letters have been converted into lower case beforehand). The length l of the shortest sub-lists for this value can be calculated as l = b4 × 0.8c = 3. Therefore, four sub-lists each containing three bigrams will be generated for this BKV: [mi,it,th], [sm,it,th], [sm,mi,th], and [sm,mi,it]. Each of these is generated by removing one of the four original bigrams. These sublists will then be converted back into strings to form the actual key values used in the inverted index, as is shown in Figure3 The identifier of the record R1will be inserted into the five inverted index lists with key values 'smmiitth', 'miitth', 'smitth', 'smmith', and 'smmiit'. With an even lower threshold (t < 0.75), sublists of length two would be generated recursively from the sub-lists of length three.

Number of bigrams in BKVs, k	Threshold value t			
	0.9	0.8	0.7	0.6
3	4	4	4	7
4	5	5	11	11
6	7	22	22	42
8	9	37	93	163
10	11	56	176	386
12	79	299	794	1586
15	121	576	4944	9949
20	211	6196	60,460	263,950

Table 1: Number of bigram (q = 2) sub-lists (s) generated according to Equation 10 for different threshold values t and different number of q-grams k in the BKVs.

The number of sub-lists generated for a BKV depends both upon the number of q-grams it consists of, as well as the value of the threshold t. Lower values of t will lead to an increased number of shorter sub-lists, and therefore a larger number of different index key values. The longer a BKV is, the more sub-lists will be generated. For a BKV of length c characters, there will be k = (c − q + 1) q-grams, and with l = max (1, bk × tc) the length of the shortest sub-lists, a total of

$$s = \sum_{i=l}^{k} \binom{k}{i}$$

Sub-lists will be generated from this BKV. From Table 1 it can be seen that the value of s grows exponentially with longer BKVs, and as the threshold t is set to lower values. The time required to generate the q-gram sub-lists will therefore be dominated by the recursive generation of sub-lists for longer BKVs.

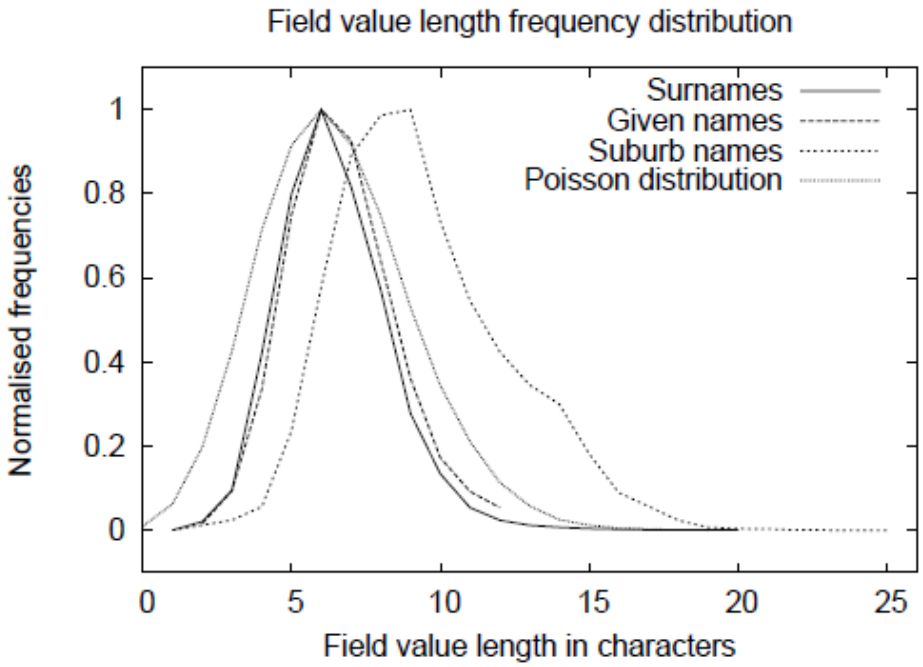


Figure 3.1.2 Normalized length frequency distributions of record field values from an Australian telephone directory containing around seven million records.

Above figure shows the length frequency distributions of values from three record fields that are common in many databases that contain information about people. As can be seen, these distributions roughly follow Poisson distributions with parameter 5 _ _ _ 10. Therefore, assuming BKV lengths that follow a Poisson distribution, it is possible to estimate the overhead of q-gram based indexing compared to traditional blocking. Let v denote the average number of times each record identifier is inserted into an inverted index list (block), compared to being inserted into just one index list as is done with traditional blocking (i.e. v = 1).

3.2 Finding approximate q-Grams

In this section we focus on the problem of finding all the text q-samples that appear inside a given pattern block Qi, that is, find all the indexes r such that dr ∈ Uq e (Qi). The first observation is that it is not necessary to generate the whole Uq e (Qi), since we are interested only in the q-samples that appear in the text (more specifically, in their positions). So we actually generate

$$I_e^q(Q_i) = \{r \in 1 \ldots \lfloor n/h \rfloor, bed(d_r, Q_i) \leq e\}$$

The idea is to store all the different text q-samples in a trie data structure, where the leaves store the corresponding r values. A backtracking approach is used to find all the leaves of the trie that are relevant for a given pattern block Qi, that is, those that match inside Qi with at most e errors.

From now on we use Q = Qi and free variable i for other purposes. If considering a specific text q-sample S = s1 . . . sq (corresponding to some dr), the problem is solved by the use of the dynamic programming algorithm

explained in the Introduction, where the text is the pattern block Q and the pattern is the text q-sample S. That is, we fill a matrix $C_{0...q,0...|Q|}$ such that $C_{i,\ell}$ is the smallest edit distance between $S_{1...i}$ and a suffix of $Q_{1...\ell}$. When this matrix is filled, we have that the text q-sample S is relevant if and only if $C_{q,\ell} \leq e$ for some ℓ (in other ords, S matches somewhere inside Q with at most e errors). In a trie traversal of the q-samples, the characters of S are obtained one by one as we descend by each branch, so this matrix will be filled row-wise rather than column-wise, which is the typical choice in on-line searching.

4. ALGORITHM AND PARAMETERS OF THE ALGORITHM

To provide efficient record linkage and to eliminate duplicated data here we use q-gram indexing techniques which two individual algorithms for indexing and searching as in the below.

Indexing (T1...n)

1. Choose q and h, $q \leq h$
2. Initialize empty trie of q-samples
3. For $r \in 1 \ldots \lfloor n/h \rfloor$
4. Insert $dr = T_{h(r-1)+1...h(r-1)+q}$ into the trie

Searching (P1...m, k)

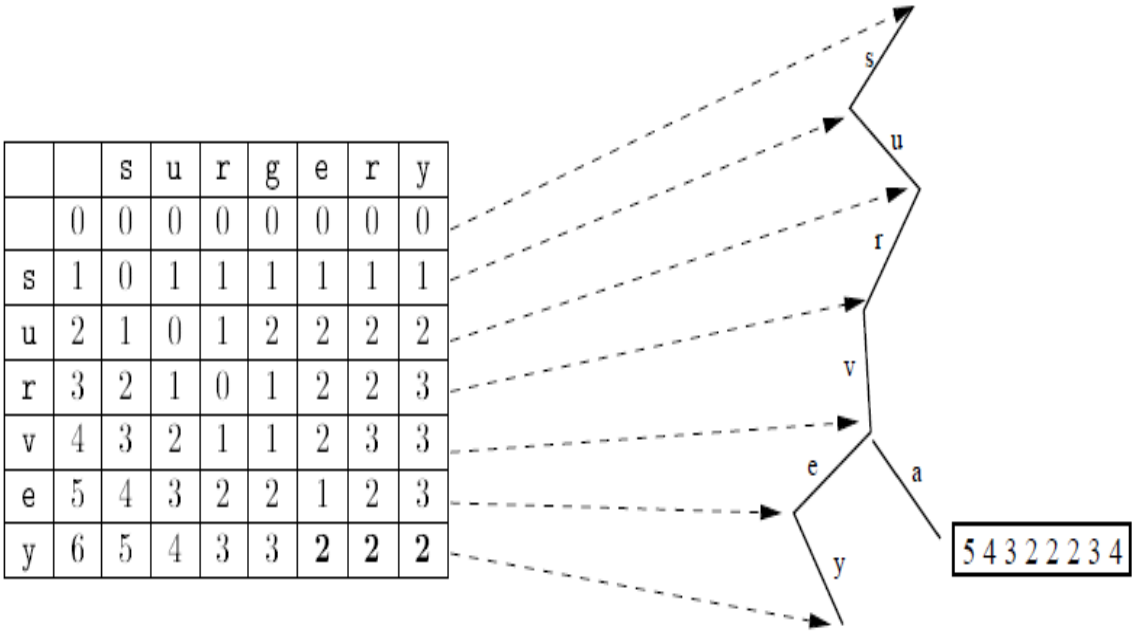
1. Choose j, $1 \leq j \leq \lfloor m-k-q+1h \rfloor$
2. Choose e, $\lfloor k/j \rfloor \leq e < q$
3. For $r \in 1 \ldots \lfloor n/h \rfloor$
4. $Mr \leftarrow j(e + 1)$
5. For $i \in 1 \ldots j$
6. $Qi \leftarrow P_{(i-1)h+1...ih+q-1+k}$
7. For each trie q-gram dr such that $bed(dr,Qi) \leq e$
8. $Mr+j-i \leftarrow Mr+j-i - (e + 1) + bed(dr,Qi)$
9. For $r \in 1 \ldots \lfloor n/h \rfloor$
10. If $Mr \leq k$
11. Run dynamic programming over $T_{h(r-j-1)+2...h(r-j-1)+m+k+1}$

Indexing and searching pseudo code, at a conceptual level. In lines 1 and 2 of the search process it may happen that no suitable j or e value exists, in which case the index is not suitable for that (q, h,m, k) combination.

4.1 The Parameters of the Algorithm

The value of e has been left unspecified in the previous development. This is because there is a tradeoff involved. If we use a small e value, then the search for the e-environments will be faster, but as we have to assume that the text q-samples not found have only e+1 errors (which may underestimate the real number of errors they have), some unnecessary verifications will be carried out. On the other hand, using larger e values gives more exact estimates of the actual number of errors of each text q-sample and hence reduces unnecessary verifications, in exchange for a higher cost to find the e-environments.

As the cost of this search grows exponentially with e, the minimal e = ⌊k/j⌋ can be a good choice. With the minimal e the sequences Dr−j+i are assumed to have j(⌊k/j⌋ + 1) errors, which can get as low as k + 1. In that particular case we can avoid the use of counters, since every text q-gram dr−j+i found inside Qi will trigger verification in Dr.



		s	u	r	g	e	r	y
	0	0	0	0	0	0	0	0
s	1	0	1	1	1	1	1	1
u	2	1	0	1	2	2	2	2
r	3	2	1	0	1	2	2	3
v	4	3	2	1	1	2	3	3
e	5	4	3	2	2	1	2	3
y	6	5	4	3	3	**2**	**2**	**2**

Figure 4.1.1 The dynamic programming algorithm run over the trie of text q-samples.

It is interesting to consider the interplay between the different remaining parameters h, q and j. Eq. (1) relates these parameters, introducing also m and k in the condition. For a given query, j has a maximum acceptable value. As j grows, longer test sequences with less error per sample are used, so the cost to find the relevant q-samples decreases but the amount of text verification increases.

So j and e permit finding the best compromise between both parts of the search. On the other hand, q and h determine the space usage of the index, which is in the worst case O(σq + n/h) integers (one header per different q-sample and one integer per sampled text position). Having a smaller index restricts more the allowed j values and, indirectly, the e values.

5. CONCLUSION

The indexing techniques presented in this survey are heuristic approaches that aim to split the records in a database (or databases) into (possibly overlapping) blocks such that matches are inserted into the same block and non-matches into different blocks. In this section we describe our actual index implementation and evaluate its

performance. Our implementation is rather simple and in-memory, omitting several possible improvements. Our trie of q-grams is implemented as a tree with pointers.

The text positions of all the q-samples are stored as plain integers without compression, in a single chunk of memory (the construction makes two passes over the text to recompute the sizes to be allocated to each q-sample inside the large chunk). While future work in the area of indexing for record linkage and deduplication should include the development of more efficient and more scalable new indexing techniques, the ultimate goal of such research will be to develop techniques that generate blocks such that it can be proven that all comparisons between records within a block will have a certain minimum similarity with each other, and the similarity between records in different blocks.

ACKNOWLEDGEMENT

We would like to thank the reviewers and also thank the comments of an anonymous referee that improved the readability of the paper.

REFERENCES

[1] G. Gonnet, R. Baeza-Yates, and T. Snider. Information Retrieval: Data Structures and Algorithms, chapter 3: New indices for text: Pat trees and Pat arrays, pages 66–82.

[2] D. E. Clark, "Practical introduction to record linkage for injury Research," *Injury Prevention*, vol. 10, pp. 186–191, 2004.

[3] 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.

[4] W. E. Winkler, "Overview of record linkage and current research directions," US Bureau of the Census, Tech. Rep. RR2006/02, 2006.

[5] W. E.Winkler, "Methods for evaluating and creating data quality," *Elsevier Information Systems*, vol. 29, no. 7, pp. 531–550, 2004.

[6] C. Faloutsos and K.-I. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *ACM SIGMOD'95*, San Jose, 1995, pp. 163–174.

[7] C. C. Aggarwal and P. S. Yu, "The IGrid index: Reversing the dimensionality curse for similarity indexing in high dimensional space," in *ACM SIGKDD'00*, Boston, 2000, pp. 119–129.

[8] N. Adly, "Efficient record linkage is using a double embedding scheme," in *DMIN'09*, Las Vegas, 2009, pp. 274–281.

[9] P. Christen and A. Pudjijono, "Accurate synthetic generation of realistic personal information," in *PAKDD'09, Springer LNAI*, vol. 5476, Bangkok, Thailand, 2009, pp. 507–514.

[10] T. de Vries, H. Ke, S. Chawla, and P. Christen, "Robust record linkage blocking using suffix arrays and Bloom filters," *ACM TKDD*, vol. 5, no. 2, 2011.

[11] M. Weis, F. Naumann, U. Jehle, J. Lufter, and H. Schuster,"Industry-scale duplicate detection," *Proceedings of the VLDB En- dowment*, vol. 1, no. 2, pp. 1253–1264, 2008.

[12] M. Bilenko, B. Kamath, and R. J. Mooney, "Adaptive blocking: Learning to scale up record linkage," in *IEEE ICDM'06*, Hong Kong, 2006, pp. 87–96.

[13] M. Michelson and C. A. Knoblock, "Learning blocking schemes for record linkage," in *AAAI'06*, Boston, 2006.

[14] D. Dey, V. Mookerjee, and D. Liu, "Efficient techniques for online record linkage," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 3, pp. 373–387, 2010.

[15] G. V. Moustakides and V. S. Verykios, "Optimal stopping: A record-linkage approach," *Journal Data and Information Quality*, vol. 1, pp. 9:1–9:34, 2009.

Author Profile

Dr. M.V.Siva Prasad, Principal of Anurag Engineering College He received B.E. [CSE] from Gulbarga University, M.Tech. [SE] from VTU, Belgaum and He was awarded Ph.D from Nagarjuna University, Guntur. He has published number of papers in International & National journals.He is a Life member of ISTE M.No. : LM 53293 / 2007. His research interests are information security, Web services, mobile computing, Data mining and Knowledge.

CH.Krishnaprasad received Master of Technology (Computer Science Engineering) from JNTU-H. His research interests are information security, Web services, mobile computing, Data mining and Knowledge.

B.Rambabu pursuing Master of Technology (Computer Science and Engineering from JNTU-H), he received B-tech.[IT] from JNTU-H. His research interests are Data mining and knowledge, Cloud Computing, Webservices.

