Improving Near Duplicate Data Detection via DSound Phonetic Matching Algorithm: A Solution to Address Typographical Problems

Cihan Varol and Sairam Hari
Department of Computer Science
Sam Houston State University
email: { cxv007@shsu.edu, sxh020@shsu.edu }

***Abstract*—Near duplicate data not only increase the cost for information processing, but also increase the time taken for a decision. Therefore, detecting and eliminating them is vital for business decisions. Shingling algorithm has been used in detecting near duplicates in large-scale text databases. The algorithm is based on the number of common tokens in two or more set of information. In other words, if there is a slight variation of the text, such as misspelling, in one of those documents, the performance of the algorithm decreases. Therefore, in this work, we proposed to embed a new phonetic approximate algorithm, namely DSound, to Shingling algorithm for improving the near duplicate data detection if there is a typographical error. Based on the experiments on real dataset, this newly proposed framework improved the Shingling algorithm's performance by 16 percent.**

Keywords-data cleansing; data quality; duplicate detection; DSound; Shingling

I. INTRODUCTION

Now-a-days, duplicate document is a common problem in big database. Advanced duplicate detection techniques are required not only to process query, but also to filter the redundant (duplicate data) information in large scale document database to improve the search quality. To address this issue, duplicate document detection techniques are used to prevent the search results from including the multiple documents having the same or nearly same content.

The search quality in a data is affected as a result of multiple copies being included in the search results. The process of indexing the data is done by scanning the content of each and every document. When two documents contain identical content, they are regarded as duplicates. There might be some documents with small dissimilarities and are not declared as being "exact duplicates" of each other but are identical to such an extent that they can be declared as near-duplicates [1]. Detecting near duplicates are utmost important to improve the search quality.

Following are examples of near duplicate samples seen in documents [2]:

- Documents containing a few different words - widespread form of near-duplicates.
- Documents with the same content but different formatting – for instance, the documents may contain the same text, but dissimilar fonts, bold type, or italics.
- Documents with the same content but with typographical errors (mistyped words)
- Plagiarized documents and documents with different versions
- Documents with the same content but different file type – for instance, Microsoft Word and PDF.
- Documents providing identical information written by the same author being published in more than one domain.

Relying only on the resemblance of the exact contents of the documents may yield to overlook near identical ones. Thus, this will eventually lead to miss the detection of duplicate documents. Although, all the listed examples are part of the near duplicate problem, in this work, we will particularly address the problems raised with *typographical errors*. *Typographical errors* are a spelling error that can be captured simply because it is mistyped or misspelled [3]. As the name suggests, these are invalid strings, properly identified and isolated as incorrect representations of a valid word [4]. Fat fingering places an important role in this type of misspelling. These errors are made assuming that the writer or typist knows how to spell the word, but may have typed the word hastily resulting in an error [5]. No matter how the word is misspelled, this results in decrease in quality of the document and yield to near identical documents. Therefore, in this paper, we introduce a hybrid approach to improve the widely known near duplicate detection algorithm, namely Shingling algorithm. Particularly we embed phonetic matching technique to Shingling algorithm to improve the outcome of near duplicate detection.

In Section 2, we discuss related work about near duplicate detection. In Section 3, the newly proposed approach will be elaborated. In Sections 4 and 5, the test cases and experimental results will be discussed. At the end, the paper will be finalized with the conclusion section.

II. RELATED WORK

There are few techniques that have been developed to identify near duplicate documents [6]-[10], web page

 ISBN: 978-1-61208-408-4

duplicates [11]–[15], and duplicate database records [16]-[17]. Brin et al. has proposed a Copy Protection System (COPS) system to protect important and intellectual property of digital documents [6]. As a part of the Stanford Digital Library project Kumar et al. developed Stand Copy Analysis Mechanism (SCAM) to identify similar documents in online library [7]. Author in [12] proposed shingling algorithm which keeps the sketch of shingles of each document to compute the resemblance of two documents. Any documents with at least one common shingle are examined and checked whether to see if it exceeds the threshold for resemblance. Broder's shingling method is implemented in AltaVista search engine for duplicate document detection [12]. Lyon et al. investigated the theoretical background to automate plagiarism detection [8]. They observe that independently written texts have a comparatively low level of matching trigrams. The Ferret plagiarism system counts the matching trigrams of a pair of documents [8]-[9]. In [10], the authors offered a new filtering technique by exploiting the ordering information. With this way, they drastically reduced the candidate sizes and hence improved the efficiency of search.

The authors in [11] proposed two approaches to compute near duplicates between all web documents simultaneously. Both of the approaches assume that two documents *di* and *dj* can be near duplicates only when document *di* and *dj* share more than 'm' fingerprints, where 'm' is a predefined threshold [11]. Das et al. proposed a Term Document Weight (TDW) matrix based algorithm with three phases, rendering, filtering, and verification, which receives an input web page and a threshold in its first phase, prefix filtering and positional filtering to reduce the size of record set in the second phase and returns an optimal set of near duplicate web pages in the verification phase by using Minimum Weight Overlapping (MWO) method [15]. In [18], the author showed that rounding algorithms for Linear Programming (LPs) and Semidefinite Programming (SDPs) used in the context of approximation algorithms can be viewed as locality sensitive hashing schemes. Simhash is a fingerprint technique that holds the property that the two documents are near duplicate only if the fingerprints of the documents are identical, since near-duplicates differ only in a small number of bit positions [13] and [18].

As stated above and discussed in [19], most of the techniques are looking for the resemblance of the documents via the exact similarity level or use some form of approximation technique to detect the near similar documents. If the latter approach is used in the whole document the computation overhead will be high. If the first approach is used alone, then the misspelled or slightly incorrect representation will yield to decrease the accuracy rating. Therefore, there is a dire need to combine these two main approaches in a way that it will not only improve the accuracy rating, but also keeps the computational overhead in low levels.

III. METHODOLOGY

In this work, we propose to embed DSound Phonetic Approximate algorithm to Shingling algorithm for detecting near duplicates documents.

Shingling algorithm is a well-known technique used for detecting the near duplicates. DSound is a newly designed phonetic based spelling correction algorithm which returns an equivalent word if there is a misspelling. Both of these methods are explained briefly in below sections.

A. Shingling Algorithm

Shingling algorithm is a well-known technique introduced by Broder to estimate the degree of similarity among pairs of documents [12]. The algorithm does not depend on any linguistic knowledge other than the ability to tokenize documents into a set of words based on the shingle size [12]. In shingling, the string is divided into words and all word sequences of adjacent words are extracted. If two documents contain the same set of shingles they are considered identical and if their sets of shingles appreciably overlap, they are accepted as exceedingly identical [12].

In detail, shingling divides the text document into set of substrings of length *k* (called *k-shingles*). The shingling algorithm is implemented using sliding window method with the window size of *k (shingle size of k)* over the input document character by character and placing the substrings into a set. For example, the string "abcabcac" can be represented by {"abc", "bca", "cab", "abc", "bca", "cac"} if the sliding window is chosen as 3-shingles. Since the algorithm eliminates the duplicate ones, only four shingles {"abc", "bca", "cab", "cac"} will be in the final set. Another example for shingling algorithm is given below:

Given a string S = {a rose is a rose is a rose}
Using shingling algorithm (with window size set at 4) the string can be tokenized as {(a rose is a), (rose is a rose), (is a rose is), (rose is a rose), (a rose is a), (rose is a rose)}.
Removing the duplicates will yield to the following shingles
{ (a,rose,is,a), (rose,is,a,rose), (is,a,rose,is) }

At the end, Jaccard similarity is used for expressing similarity of sets. For sets A and B, it is defined in (1).

$$SIM(A, B) := |A \cap B| / |A \cup B|. \quad (1)$$

The result ranges from 0 (no elements in common) to 1 (identical). It is evident that duplicate documents have a higher similarity to the original than unrelated documents.

B. DSound Phonetic Matching Algorithm

Phonetic matching algorithms are relied and dependent on the phonetic structures of a language. In spite of this restriction, they are proved to be very effective when fixing ill-defined data in English language [20]. Moreover, computation wise, it can be faster than string matching algorithms because of the applied conversion rules can shrink the textual information to a small number of digit

coding. Because of the stated reasons, in this work, we have concentrated on phonetic matching algorithms.

Although it is one of the early designed algorithms for phonetic matching, previously we have shown the effectiveness of Soundex algorithm [20]. The Soundex algorithm keeps the first letter in a string and converts the rest into numbers. All zeros (vowels and 'h', 'w' and 'y') are then removed and sequences of the same number are reduced to one only (e.g., '222' is replaced with '2'). The final code is the original first letter and three numbers (longer codes are cut-off, and shorter codes are extended with zeros). As an example, the Soundex code for 'Rodgers' is 'r326', and 'r262' for 'Rojers'.

Although Soundex is a well-designed solution, a major problem with the algorithm is that it keeps the first letter, thus any error at the beginning of a name will result in a different Soundex code. This can cause to eliminate the valid candidates because of the error in the first letter. Another disadvantage of the algorithm is the lack of transformation rules for special conditions, such as a word containing –dge- letters consecutively, i.e., Rodgers. As shown in the above examples with 'Rodgers' and 'Rojers', although those two text information are very close to each other, the difference between the codings are large enough to not easily come to a conclusion that those text information can be same. Moreover, truncated the coding with only 3 digit and letter coding may yield to miss misspellings occurring late in a long word. Therefore, we have created DSound phonetic matching algorithm based on the deficiencies of the Soundex and created three steps to generate a phonetic coding for a given text information as shown in Table 1.

TABLE I. DSOUND TRANSFORMATION RULES

Step	Letters	Code
1	kn-, gn- pn, ac-, wr-	drop first letter
	x-	change to "s"
	wh-	change to "w"
	otherwise	skip to step 2
2	d	2 (if in –dge- or –dgi- 'Rodgers') – drop 'g'
	g	0 (if in –gh- 'Houghton')
	t	2 (if in -tia- or -tio- 'Attention')
	otherwise	skip to step 3
3	a e h i o u w y	0
	b f p v	1
	c g j k q s x z	2
	d t	3
	l	4
	m n	5
	r	6

In detail, first the initial characters are parsed by the given rules. Second, special cases are handled based on the created conversion system. Third, as a last step, the final text information is converted to digits based on the given rules and only repeated digits are removed from the final coding. For instance, the DSound code for "Rodgers" and "Rogers" are same and will yield to a coding of '602062'

C. Proposed Approach

The proposed approach is a combination of the shingling algorithm and DSound phonetic matching algorithm as shown in Figure 1.

Specifically, two input files are preprocessed and inserted into the system to check if there are any inconsistencies (mistyped words) between those. If there is any, then the proposed approach utilizes the DSound phonetic matching algorithm for finding the closest match for the mistyped data with the help of dictionary, which holds the English words. If there is no perfect match (different DSound codings), then Edit Distance score [21] is calculated between the original data and suggestions. In order to eliminate irrelevant results from the suggestion list a threshold of 2 is set for the distance. Then the closest candidate is selected as the best possible replacement for the mistyped word and is replaced with the ill-defined one in the text document. After the mistyped words are modified according to the combined approach, Shingling algorithm is applied to see the degree of the closeness of the text files. The closeness is calculated by the number of common shingles divided by the total number of shingles in both files. The details of the algorithm are provided in the following pseudo code (Figure 2).

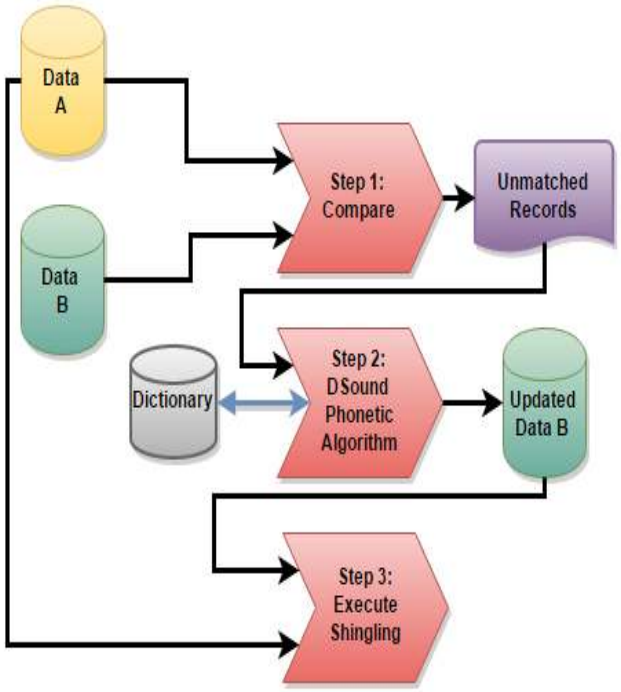

Figure 1. Proposed Hybrid Approach

Algorithm

```
1:  Original File = OF;
2:  Reference File = RF;
3:  Open OF and RF
4:  Run Shingling Algorithm
5:  Compare OF and RF
6:  Run DSound Algorithm on not-matched results
7:      IF: match found = 1
8:          write and replace the word in RF
9:      ELSE
10:         calculate edit distance
11:         write and replace the word in RF with the
              (lowest score) closest one
12:     END IF
13: Open OF and RF
14: Run shingling algorithm
15: Compare OF and RF
```

Figure 2. Pseudo code of the Hybrid Approach

IV. TEST CASE AND RESULTS

The data sets used for testing the algorithm were open to public text files that were obtained from the internet. Two files were obtained for each data set. The first file contained the error-free version of the text, while the second one held the same content but with occasional typographical errors due to mistyping or OCR errors. Overall, two files were duplicates, but because of some mistyped words the degree of closeness was less. The length of the text varied between 119 words to 1108 words. As a proof of concept we executed the proposed approach on the mistyped version of the Abraham Lincoln's Gettysburg address [22]. First, the mistyped words were corrected using the DSound phonetic matching algorithm. Second, Shingling algorithm was applied to the updated data set to evaluate the algorithms' performance (Table II).

TABLE II. FIRST TEST DATA RESULTS

	Shingle size=3	**Shingle size=4**	**Shingle size=5**
Common Shingles w/o Proposed Approach	254	252	247
Not Matching Shingles w/o Proposed Approach	26	34	42
Common Shingles with Proposed Approach	265	263	262
Not Matching Shingles with Proposed Approach	7	11	12

As shown in Table II, before applying the hybrid approach and given a shingle size of 3, the number of common shingles between the two input files was 254 and the number of un-matched shingles was 26, after duplicate shingles were removed. After we applied the hybrid approach to the Shingling algorithm, the number of common shingles increased to 265 and the number of un-matched shingles dropped to 7 with the shingle size of 3, once again after duplicate shingles were removed. The difference between the total number of shingles produced before using the proposed approach and after using the proposed approach is because of the removal of duplicates from the final set of shingles. As reflected from the results, the hybrid approach improved the performance of shingling algorithm and increased the degree of closeness between the two input files with different shingle sizes.

As a second task, we executed the same test on forty documents. Overall, 40 pairs of error free and mistyped versions of the documents were collected from the internet. The error free versions of the documents are used as a reference point and mistyped versions are compared with it. These files found in the internet were mostly because of Optical Character Recognition (OCR) problems caused by scanners [23]-[24]. As shown in Figure 3, the percentage of common shingles to all shingles was improved with an average of 16 percent with different shingle sizes of 3, 4 and 5. Although the average common shingle sizes were increased all over the board, the highest percentage of the common shingles were always seen when the shingle size was selected as 3.

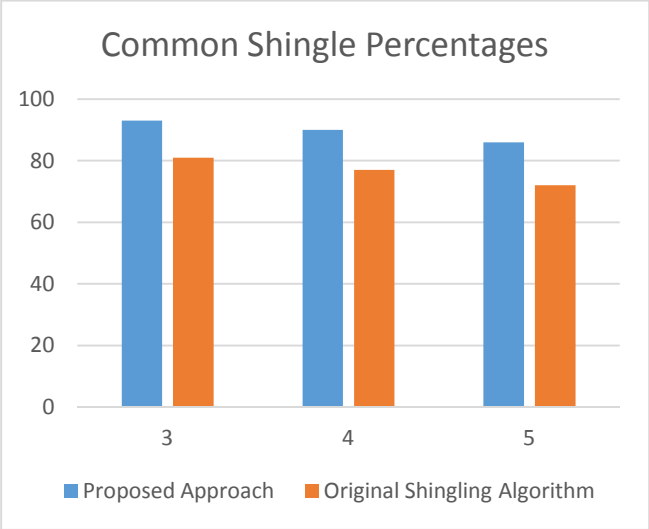

Figure 3. Percentages of Common Shingles with Different Shingle Sizes

The answer to which shingle size is better suited for detecting the near duplicates is vital while applying the Shingling algorithm. Based on the tests we conducted, our observations are reflected in Table III. Overall, the shingle size 3 provides the highest degree of closeness among two near identical documents and less overhead (total number of shingles) in all test data we experimented on. However, as expected if the all misspelled words are corrected, then the accuracy percentage would be same among different shingle sizes and the computation overhead would be very close to each other. Other shingle sizes, such as greater than 5 or less than 3 is omitted, since those window sizes yield to lower shingle scores or require extensive computation.

 ISBN: 978-1-61208-408-4

TABLE III. SELECTION OF SHINGLE SIZE

Scenario	Suited shingle size
When no misspelled words are corrected by Jaro.	3
When only some misspelled words are corrected by Jaro	3
When all misspelled words are corrected by Jaro.	3, 4 or 5

It is also important to note that correctly fixing one ill-defined data with the hybrid approach in most cases yields to substantial decrease in the number of not matching shingles. The main reason is that most of the each fix yields to multiple matching shingles based on the selected window size. Specifically, if a mistyped word is fixed when the window size is 4, then this may result in increase of 4 common shingles between the two documents, unless the mistyped word is not within the first or last three words of the document.

Word similarity and document similarity have been widely studied by researchers. In these works each word or each document are expressed in a vector space, then the similarity in the vector space can be calculated. The major disadvantage of this method is the large space vector and the lack of considering relationship and order between terms. It is computationally inefficient due to the sparse sentence vector. Moreover, it doesn't do any favoring if there is any slight spelling error in the compared text. However, our proposed approach is particularly designed to address the errors caused by misspellings. That is why, we rationale our technique based on the assumption that the misspelled information is close to each other and the context and weights associated with the text is not relevant.

V. PHONETIC MATCHING ALGORITHMS

In this work, we particularly created a new phonetic matching algorithm to increase the efficiency of the near duplicate detection. To prove that the DSound algorithm performs best as the phonetic matching algorithm of this newly proposed hybrid system, we compared its efficiency with other well-known phonetic matching algorithms, such as Soundex, Phonex, Phonix, and Double Metaphone. As mentioned earlier, Soundex is used to correct phonetic misspellings with mapping a string into a key consisting of its first letter followed by a sequence of digits [25]. The encoding algorithm is very fast in practice. However, a major problem with Soundex is that it keeps the first letter, thus any error at the beginning of a name will result in a different Soundex code. Phonex [25] tries to improve the encoding quality by pre-processing names according to their English pronunciation before the encoding. All trailing '*s*' are removed and various rules are applied to the leading part of a name (for example 'kn' is replaced with 'n', and 'wr' with 'r'). Like Soundex, the leading letter of the transformed name string is kept and the remainder is encoded with numbers (1 letter, 3 digits). Phonix algorithm is an improvement for the Phonex and applies more than one hundred transformation rules on groups of letters [26]. The Metaphone algorithm is also a system for transforming words into codes based on phonetic properties [25]. Unlike Soundex, which operates on a letter-by letter scheme, Metaphone analyzes both single consonants and groups of letters called diphthongs according to a set of rules for grouping consonants and then maps groups to metaphone codes.

In order to reduce the overhead, information has to be better organized and should produce relevant results. Two major metrics commonly associated with Information Retrieval Systems are Precision and Recall [27].

Precision can be defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search as shown in (2).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (2)$$

Precision measures one aspect of information retrieval overhead for a user performing a particular search. If a search has 90% precision, then 10% of the user effort is the overhead reviewing non-relevant items. Please note that the definition of relevant documents is broadened that the suggestion algorithms provide the correct result as one of the possible candidate for the misspelled word.

Recall is different than precision. Recall can be defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents, or in other words Recall is the percentage of spelling correction rate. Recall gauges how well a system processing a particular query is able to retrieve the relevant items that the user is interested in seeing. The formula of the recall is shown in (3).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (3)$$

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score. F-score is calculated by (4).

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

The discussed five algorithms are tested on the discussed forty data set as shown in Table IV.

TABLE IV. COMPARISON OF PHONETIC MATCHING ALGORITHMS

Algorithm	Precision	Recall	F-Score
DSound	83.7%	61.8%	71.10%
Soundex	78.3%	54.3%	64.13%
Phonex	76.4%	51.8%	61.74%
Phonix	79.6%	57.1%	66.50%
Double Metaphone	79.8%	56.2%	65.95%

 ISBN: 978-1-61208-408-4

As an average, DSound phonetic matching algorithm achieved highest F-Score, Precision and Recall rates. This is the main rationale why DSound phonetic matching algorithm is selected as the part of the hybrid approach

VI. CONCLUSION AND FUTURE WORK

Shingling is an effective and efficient algorithm to detect and eliminate duplicates in large-scale short text databases. However, applying this technique to original data can result in poor ratings if the documents contain mistyped information. Therefore, in this work, a new phonetic matching algorithm, namely DSound, and Shingling algorithm are combined to detect the near duplicate documents. Experiments reflected that using the hybrid approach is an encouraging solution for large-scale short text duplicate detection and increased the performance of shingling algorithm. As a future work, the hybrid approach will be used in various domains, such as in health care data, to improve the detection of near duplicates. Moreover, we will focus on language identification on the text information so that we can create and apply language specific rules for the phonetic matching process.

REFERENCES

[1] K. J. Prasanna and P. Govindarajulu, "Duplicate and Near-duplicate documents detection," European Journal of Scientific Research, Vol.32 No.4, 2009, pp. 514-527.

[2] T. Gupta and L. Banda, "A hybrid model for detection and elimination of Near Duplicates based on Web Provenance for Effective Web Search," International Journal of Advances in Engineering & Technology. Vol. 4, Issue 1, 2012, pp. 192-205.

[3] Levenshtein Edit-Distance Algorithm, http://www.nist.gov/dads/HTML/Levenshtein.html [retrieved: March, 2015].

[4] C. Becchetti and L. P. Ricotti, "Speech Recognition: Theory and C++ Implementation". John Wiley & Sons, 1999.

[5] K. Kukich, "Techniques for Automatically Correcting Words in Text", ACM Computing Surveys, vol. 24, No. 4, 1992, pp. 377-439.

[6] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," In: Proceedings of the ACM SIGMOD Annual Conference, San Francisco, CA, May 1995, pp. 398-409.

[7] K. N. Shiva and H. Garcia-Molina, "SCAM: A copy detection mechanism for digital document,". In: Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries, Austin, Texas, June 1995, pp. 1-13.

[8] C. Lyon, R. Barrett, and J. Malcolm, "A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector," In: Plagiarism: Prevention, Practice and Policies Conference, June 2004, pp. 1-7.

[9] C. Lyon, R. Barrett, and J. Malcolm, "Plagiarism is easy, but also easy to detect," Plagiary: Cross-Disciplinary Studies in Plagiarism, Fabrication, and Falsification Vol.1, No.5, 2006, pp. 1–10.

[10] C. Xiao, W. Wang, and X. Lin, "Efficient Similarity Joins for Near-Duplicate Detection," Proceeding of the 17th International Conference on World Wide Web, April, 2008, pp 131 – 140.

[11] K. N. Shiva and H. Garnia-Molina, "Finding near-replicas of documents on the web," In: Proceedings of Workshop on Web Databases, Valencia, Spain, March 1998, pp 204-212.

[12] A. Broder, "Identifying and Filtering Near-Duplicate Documents," In: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, Montreal, Canada, June, 2000, pp. 1-10.

[13] G. S. Manku, A. Jain, and A. D. Sarma, "Detecting near-duplicates for web crawling," In: Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, Canada, May 2007, pp. 141-149.

[14] M. Henzinger, "Finding near-duplicate web pages: A large-scale evaluation of algorithms," In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, U.S.A, August 2006, pp.284-291.

[15] S. N. Das, M. Mathew, and P. K. Vijayaraghavan, "An Approach for Optimal Feature Subset Selection using a New Term Weighting Scheme and Mutual Information," Proceedings of the International Conference on Advanced Science, Engineering and Information Technology, Malaysia, January 2011, pp. 273-278.

[16] Z. P. Tian, H. J. Lu, and W.Y. Ji, "An n-gram-based approach for detecting approximately duplicate database records," International Journal on Digital Libraries, Vol. 5 No. 3, 2001, pp. 325-331.

[17] M. A. Hernandez and S. J. Stolfo, "The merge/purge problem for large databases," In: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, CA, 1995, pp. 127-138.

[18] M. Charikar, "Similarity estimation techniques from rounding algorithms," In: Proceedings of 34th Annual Symposium on Theory of Computing, Montréal, Québec, Canada, May 2002, pp. 380-388.

[19] J. P. Kumar and P. Govindarajulu, "Duplicate and near duplicate documents detection: A review," European Journal of Scientific Research, 32, 2009, 514--527.

[20] C. Varol and C. Bayrak, "Personal Name-based Pattern and Phonetic Matching Techniques: A Survey," ALAR Conference on Applied Research in Information Technology, February 13, 2009, Conway, Arkansas, USA..

[21] Levenshtein VI. "Binary codes capable of correcting deletions, insertions and reversals". Doklady Akademii Nauk SSSR 163 : 845-848, 1965, also {1966) Soviet Physics Doklady 10 : 707-710.

[22] "The Gettysburg Address", http://www2.cs.uregina.ca/~simeon2m/CS210/Assignments/testfile.txt [retrieved: March, 2015].

[23] "Electronic Text Center", http://courses.cs.vt.edu/csonline/AI/Lessons/VisualProcessing/OCRscans.html [retrieved: March, 2015].

[24] "Error Report on ScarWox", http://www.columbia.edu/acis/cria/rosenberg/sample/ [retrieved: March, 2015].

[25] L. Philips, "Hanging on the metaphone," Computer Language, 7(12), 1990, pp. 39-43.

[26] T. Gagg, "PHONIX: The algorithm," Program: automated library and information systems, 24(4), 1990, pp. 363–366.

[27] J. Davis and M. Goadrich: "The relationship between Precision-Recall and ROC curves." ICML 2006: pp. 233-240.

 ISBN: 978-1-61208-408-4